AMIGA DIGA! REVIEWED

Amazing COMPUTING™

Your Original AMIGA™ Monthly Resource

Volume 2 Number 9
US $3.50 Canada $4.50

PRODUCTIVITY & YOUR AMIGA

Programming Tutorials

Fast Directories
Soundscape #2
Modula-2

Programming Tools

Manx C
Lattice C

Hardware Reviews

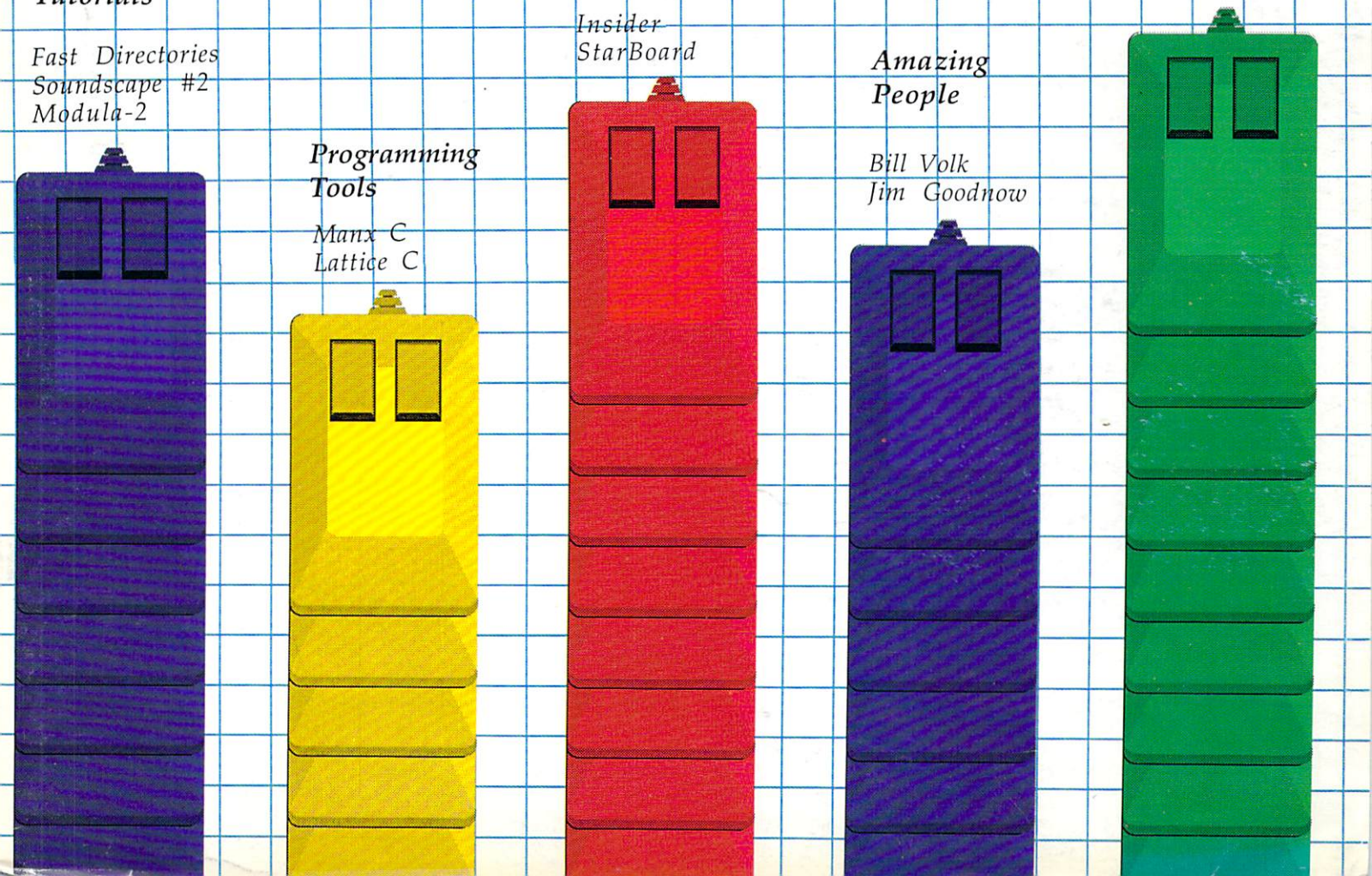
Insider
StarBoard

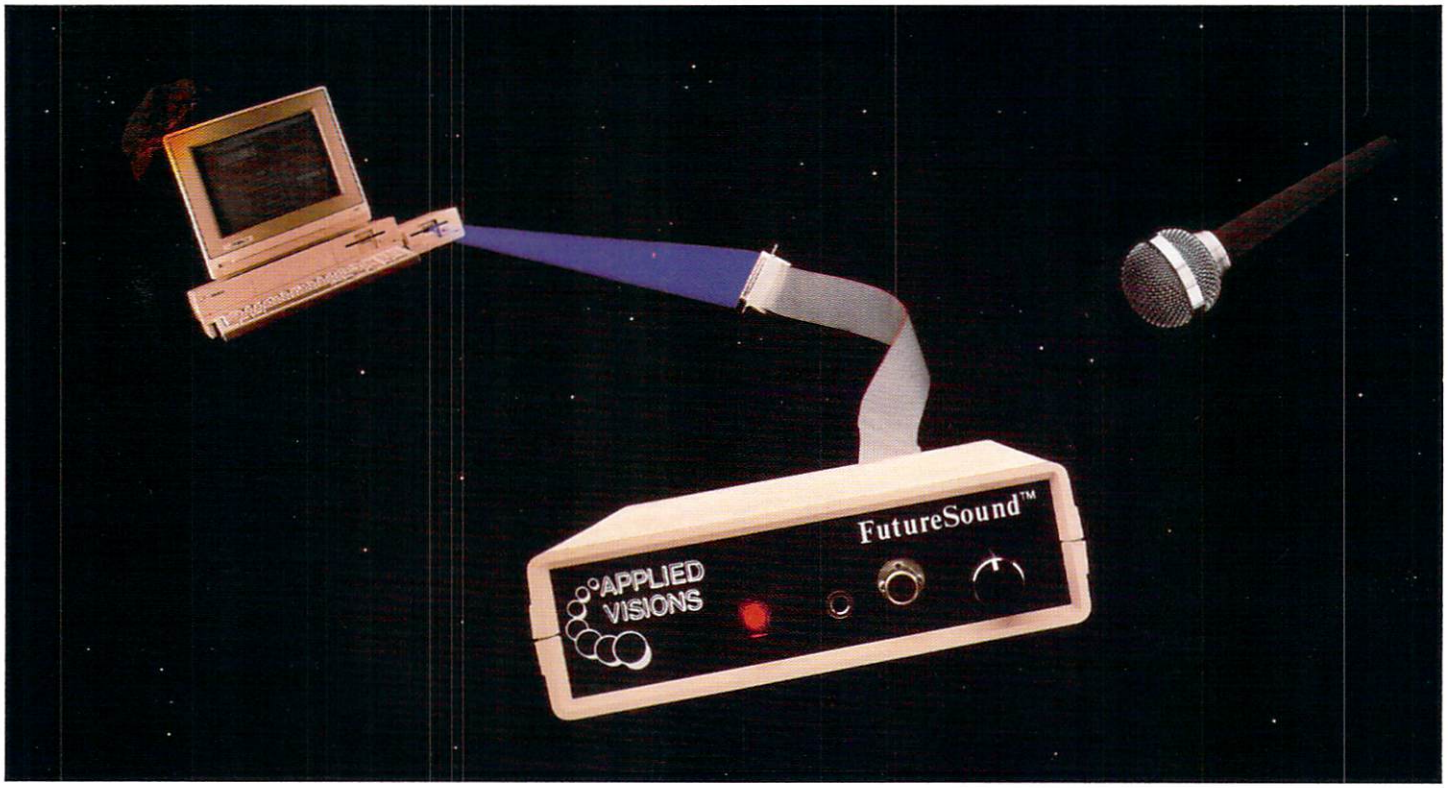
Amazing People

Bill Volk
Jim Goodnow

Business Aids

Impact
Pagesetter
Microfiche Filer
Analyze!
KickWork
Gizmoz





"Open the pod bay doors, HAL..."

Programmers cast their vote!

Right now, leading software developers are hard at work on the next generation of Amiga® products. To add the spectacular sound effects we've all come to expect from Amiga software, they are overwhelmingly choosing one sound recording package... FutureSound. As one developer put it, "FutureSound should be standard equipment for the Amiga."

FutureSound the clear winner...

Why has FutureSound become the clear choice for digital sound sampling on the Amiga? The reason is obvious: a hardware design that has left nothing out. FutureSound includes two input sources, each with its own amplifier, one for a microphone and one for direct recording; input volume control; high speed 8-bit parallel interface, complete with an additional printer port; extra filters that take care of everything from background hiss to interference from the monitor; and of course, a microphone so that you can begin recording immediately.

What about software?

FutureSound transforms your Amiga into a powerful, multi-track recording studio. Of course, this innovative software package provides you with all the basic recording features you expect. But with FutureSound, this is just the beginning. A forty-page manual will guide you through such features as variable sampling rates, visual editing, mixing, special effects generation, and more. A major software publisher is soon to release a simulation with an engine roar that will rattle your teeth. This incredible reverberation effect was designed with FutureSound's software.

APPLIED VISIONS

Question: What can a 300 pound space creature do with these sounds?

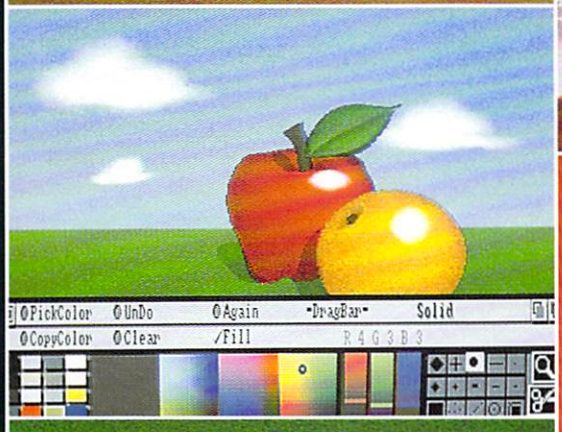
Answer: Anything he wants.

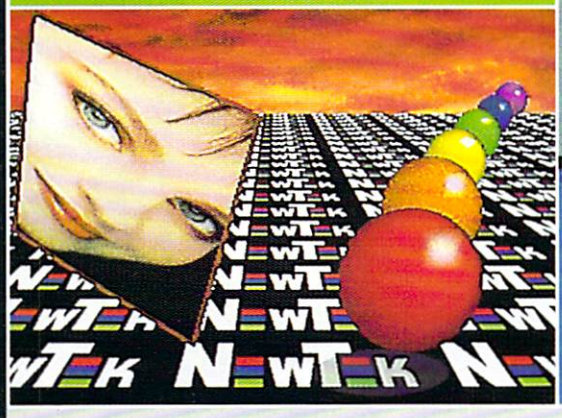
Since FutureSound is IFF compatible (actually three separate formats are supported) your sounds can be used by most Amiga sound applications. With FutureSound and Deluxe Video Construction Set from Electronic Arts, your video creations can use the voice of Mr. Spock, your mother-in-law, or a disturbed super computer.

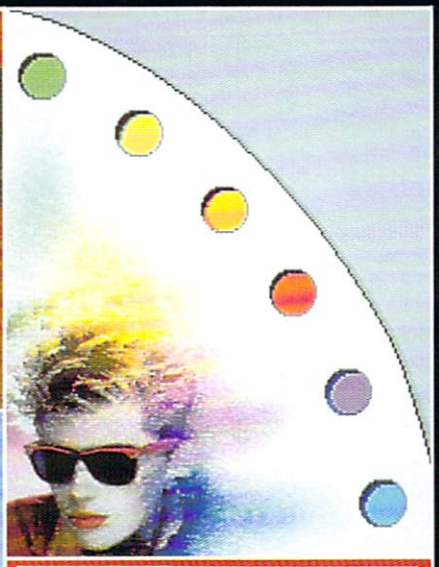
Programming support is also provided. Whether you're a "C" programming wiz or a Sunday afternoon BASIC hacker, all the routines you need are on the non-copy protected diskette.

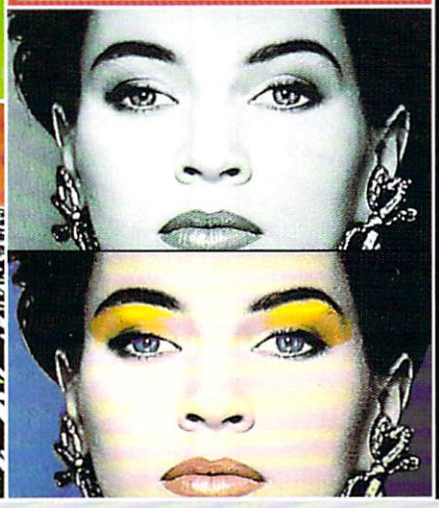
Your Amiga dealer should have FutureSound in stock. If not, just give us a call and for $175 (VISA, MasterCard or COD) we'll send one right out to you. Ahead warp factor one!

Applied Visions, Inc., Suite 2200, One Kendall Square
Cambridge, MA 02139 (617) 494-5417









ONLY DIGI PAINT CAN DO ALL THIS

Get the maximum graphics power from your Amiga. Create stunning, lifelike computer artwork with Digi-Paint, the first full-featured 4096 color (Hold and Modify) paint program. Break the "32 color barrier" and finally realize the potential of your Amiga with Digi-Paint's advanced features:

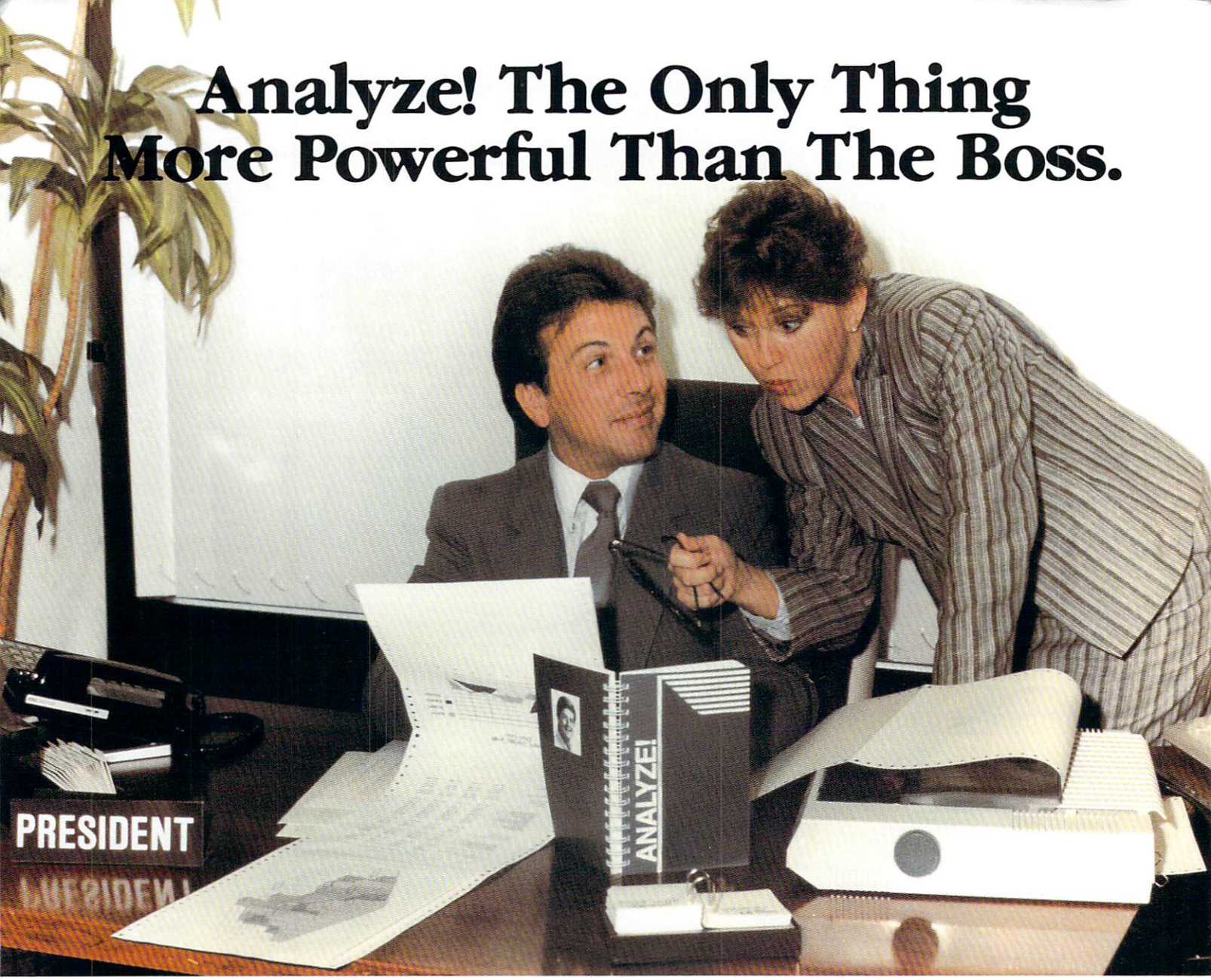
- 4096 colors on screen simultaneously
- NewTek's exclusive enhanced HAM mode
- Dithered HAM gradient fill
- Full screen effects including double, half size, mirror reverse and more
- Full IFF and Digi-View compatibility
- Use 320x200 or HAM hi-res 320x400 resolutions
- Fat bits Magnify mode
- Rectangle, oval, line and other drawing tools
- 12 different paint modes including blending, tinting and smooth shading
- Full lasso cut and paste with automatic edge blending
- Programmed completely in assembly language for fast, smooth response

Find out why Byte Magazine called Digi-Paint "Remarkable". Available now at your local Amiga dealer or call: **1-800-843-8934.**

ONLY $59.95

NEWTEK INCORPORATED

Analyze! The Only Thing More Powerful Than The Boss.



Why is **Analyze!**™ more powerful than the boss? Because like your boss it's fast, efficient and most importantly, **Analyze!** is never wrong.

Analyze! is an electronic spreadsheet program for the Commodore Amiga® that turns you and your computer into financial wizards. **Analyze!** is easy to use, yet sophisticated enough to keep track of anything, from your checkbook to your company's general ledger.

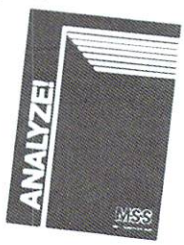
Analyze! commands can be entered from the mouse or keyboard. Dozens of built-in functions make complex calculations effortless.

Your data can be presented using one of **Analyze!'s** 8 breathtaking graph models. As many as 4 graphs in 4 or 8 colors can be displayed to create professional looking sales reports.

Additional features like a macro language to automate your spreadsheets, powerful print, sort functions and compatibility with other popular spreadsheets will make you wonder how you ever lived without **Analyze!.**

Analyze! is easy to learn, supported by an informative reference manual and our expertly trained Technical Support Division.

Ask for a dealer demonstration of this fine product, or call our Micro-Systems Software Technical Support Line at (305) 790-0772



For a dealer near you call:
Brown-Wagh Publishing:
(800) 451-0900
In CA: (408) 395-3838
16795 Lark Ave., Suite 210
Los Gatos, CA 95030

MSS
Micro-Systems Software

8 years of quality software and still growing strong!

12798 West Forest Hill Blvd.
West Palm Beach, FL 33414

Amazing Dealers

The following are Amazing Dealers, dedicated to supporting the Commodore-Amiga™.
They carry Amazing Computing™, your resource for information on the Amiga™.

If you are not an Amazing Dealer, but would like to become one, contact:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA. 02722
1-617-678-4200

Alabama
Abex Data Systems — Huntsville
Alabama Book Store — Tuscaloosa
Command Computers — Birmingham
Electronics Boutique — Hoover
Madison Books & Computers — Madison
Mail Tel Communications — Mobile
Mel's Photo & Computer Shop — Montgomery
Universal Computer System — Mobile

Arkansas
SIS Inc. — Mt. Home
The Micro Shop — Little Rock

Arizona
Computer West — Phoenix
Computer Works — Glendale
Copperstate Business Systems — Phoenix
Copperstate Computers — Tucson
Hacko's — Tempe
North American Digital — Tucson

California
Aday International — Foster City
Augnet — Freemont
Balboa Office Supply — San Diego
Bits n Pieces Comp Prod. — Anahiem
Brown Knows Computing — Redlands
Byte Factory Computers — Salinas
C.F.T. — Clairemont
Century Computer Systems — La Habra
Circus of Books II — LosAngeles
Commotech — National City
Compare Inc. — Ridgecrest
Computer Attic — Palo Alto
Computer Nook, Inc. — San Bernadino
Computer Outlet — San Diego
Computer Outlet — San Marcus
Computer Time — Citrus Hts
ComputerLand Freemont — Freemont
Dataphile — Watsonville
Delta Spectrum — Laguna Beach
De Lauer's News Agency — Oakland
Electronics Boutique — West Los Angeles
Empire Computers — Santa Rosa
Harding Way News — Stockton
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
Home Computing Center — San Bruno
HT Electronics — Sunnyvale
K J Computers — Granada Hills
Levity — North Hollywood
LTC Computers — Santa Ana
Micro Flight — San Jacinto
Monagon Software — El Cerrito
Op Amp Tech Books — Los Angeles
World Book & News — Hollywood
R&R Software — Chula Vista
R&R Software — San Diego
Ridgecrest Computer Ctr. Inc. — Ridgecrest
S.O.S. Computers — Los Angeles
Software First — Santa Rosa
Software Cellar — Anahiem
Software Plus — Citrus Heights
Tower Books — Chico
Tower Books — Citrus Hghts
Tower Books — Concord
Tower Books — Mountain View
Tower Books — Sacramento
Tower Books — San Mateo
The Computer Room — Scotts Valley
Winner's Circle — Berkley

Colorado
Book Tech Distributing — Englewood
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Room — Aurora
Ebert Personal Computers — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Sun Country Computers — Colorado Springs
Aurora Newsstand — Aurora

Connecticut
Amiga Business Computer — Norwalk
Computer Outlet — PlaVernon
Electronics Boutique — Danbury
Friendly Computers — Plantsville
Mnemonics Inc. — Norwalk
Personal Computer center — Norwich
Software City — Orange
Software Kingdom — East Windsor
Spectrum Computers — New Haven

Delaware
Castle Video — Newark

Florida
A A Computers — Jacksonville
Commodore Country Inc. — Pinellas Park
Computer Bar — Pennsacola
Computer Base — Brandon
Computer Image — Miami
Computer Specialty — Melbourne
Computer Terminal Inc. — Sebring
Computers 4 Rent — W. Palm Beach
Cotronics Inc. — Stewart
Education Computers Etc. — Tallahasee
Electronics Boutique — Boynton Beach
Electronics Boutique — Tallahassee
Electronics Boutique — Miami
Electronics Boutique — Orlando
Family Computers — Sarasota
Florida Book Store — Gainsville
Games 'N' Gadgets — Orange Park
Games 'N' Gadgets — Jacksonville
Games 'N' Gadgets — Tallahassee
Games 'N' Gadgets — Clearwater
Gulf Coast Computer Exchange — Panama City
Heartland Computers — Mulberry
K&L Computers — E. Palatka
Megaport Computer Center — Venice
Micro Computer Center — Largo
New Age Electronics — St. Petersburg
Random Access Computers Inc. — Ft. Walton Beach
The Open Door — Cocoa
The PC Collection — Coral Gables

Georgia
Electronics Boutique — Atlanta
Electronics Boutique — Atlanta
Electronics Boutique — Kennesaw
Future System — Augusta
Games 'N' Gadgets — Atlanta
Games 'N' Gadgets — Atlanta
Software South Inc. — Savannah
The 64 Store — Atlanta

Hawaii
Astro Computers & Electronics — Aiea
Computer House Inc. — Honolulu
J & C Repair — Aiea

Idaho
A.B.J. Computer & Video — Boise
A.B.J. Computer & Video — Nampa

Illinois
A Bitketer — Rockford
Computerland O'Hare — Park Ridge
ComputerLand of Niles — Niles
DigitalWorld — Addison
I.C.V. Data Center Inc. — Mundelein
Elini Microcomputers Inc. — Naperville
Interstate Computer Service — LaMont
Kroch's & Brentano's — Chicago
Micro Age — Oakiawn
Software Or Systems — Fairview Hts.
Software Plus — Wheeling
Software Plus West — Hanover Park
The Memory Expansion — Normal
Unique Computer — Sterling

Indiana
Burkat Computer Ctr. — South Bend
Bytrox Computer — Ft. Wayne
Bytrox of Angola — Angola
Computer Corner — Fort Wayne
Computer People Inc. — Michigan City
Greg Chaney — Greenfield
Micro Computer Centers Inc. — Indianapolis
Spectra Computer Sys. Inc. — Evansville
Vons Computers — West Lafayette

Iowa
Gronet Computers — Des Moine
Micro Computer Applications LTD — Marshalltown

Kansas
J & L Electronics — Liberal
MidKansas Computers — Newton
Midwest Computers — Manhattan
Theobred Computers — Topeka

Kentucky
Computer Dimensions Inc. — Lexington
Reality World Computers — Lexington
The Software Source — Louisville

Louisiana
Computer Time — Shreveport
Electronics Boutique — Kenner
Electronics Boutique — Monroe
Electronics Boutique — Laurencoville
Modern Business Machines — Metairie
Software Center International — Metairie
Software Solutions — Baton Rouge
Software Source — Lafayette
The Computer Clinic — Lafayette

Maryland
Buried Treasure — Rockville
Computer Crafters — Wheaton
Computerware TODAY — Gaithersburg
Compuvision Vision Center — Columbia
Electronics Boutique — Baltimore
Electronics Boutique — Baltimore
Electronics Boutique — Anne Arundel County
Electronics Boutique — Owings Mills
Games 'N' Gadgets — Baltimore
Games 'N' Gadgets — Baltimore
Games 'N' Gadgets — Columbia
Layhill Newsstand — Kensington
Micro Computer Center — Baltimore
Software Advantage — Rockville
Software N Things — College Park
Waldorf Computer — Waldorf

Massachusetts
Club Computer — Cambridge
D-Five — Bedford
Electronics Boutique — Holyoke
General Computer Store — Framingham
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
The Bit Bucket — West Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield

Maine
Computer Barn — Damariscotta

Michigan
Basic Computer Center — Grand Rapids
Bits-Bytes-Nibbles — Petoskey
Canton Computer — Canton
CompuSoft — Kalamazoo
Direct Access — Lansing
Edmunton Computer Center — Pontiac
Galaxy Computers — Allen Park
Global Computer Center — Dearborn
I O Software — Flint
Michigan Software — Novi
Midland Computer Shop — Midland
Central Park Computers — Okemos
Pro-Video — Okemos
Programs Unlimited — Grand Rapids
Roseville Computer Store — Warren
Slipped Disk — Madison Heights
Slipped Disk — Southfield
Software Plus — Grand Rapids
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Soft Warehouse — Grand Rapids
The Soft Warehouse — Kalamazoo
The Soft Warehouse — Portage
Ye Olde Computer Shoppe — Ypsilanti

Minnesota
Computers, etc. — Eagan
Computer Outfitters — Roseville
Computer Place — Minneapolis
JMH Software — Maple Grove
MCD of Hibbing Inc. — Hibbing
Midwest Computer Services — Clearwater
OnLine Computers — Minnetonka
Specialists IN — Hopkins
Valiant Inc. — Stillwater

Mississippi
Electronics Boutique — Ridgeland
Enterprises Unlimited — Jackson
National Computer Center — Ocean Springs

Missouri
Associated Computer Services — Springfield
Brands Mart Computers — Kansas City
Data Byte Computer Co. — St. Louis
Instant Replay Ltd. — Chesterfield
Systems Plus — Independence
Systems Plus of St. Louis — St. Louis

Montana
Appelgren Computer Systems — Great Falls

Nebraska
BULLS-EYE — Lincoln
My Com Inc. — Plattsmouth
Double E Electronics — Omaha

Nevada
Century 23 Inc. — Las Vegas
Computer World — Las Vegas
ComputerLand of Carson — Carson City
Tower Records — Las Vegas

New Hampshire
Computer Mart of New Hampshire — Nashua
Diversified Computers — Keene
Electronics Boutique — Nashua

New Jersey
Electronics Boutique — Voohrees
Electronics Boutique — Wayne
Electronics Boutique — Rockaway
Electronics Boutique — Cherry Hill
Electronics Boutique — Woodbridge
Family Computer Centres — South Orange
Games 'N' Gadgets — Burlington
Games 'N' Gadgets — Livingston
Morris County Stationers — Chester

New Mexico
New Horizon Computer System — Alamogordo
Page One Newstand — Albuquerque
Technological Concepts — Las Cruces

New York
Amuso — New York
Amicus- Hudson Valley — Wappingers Falls
Amiga Business Computer — North Port
Byte Shop — Merrick
Compurite Center — White Plains
Computer Outlet — Jamestown
Computer Palace — Felden
Cromland Inc. — Brooklyn
Electronics Boutique — Lake Grove
Electronics Boutique — Huntington Station
Electronics Boutique — Albany
Electronics Boutique — Valley Stream
Games 'N' Gadgets — Nanuet
Games 'N' Gadgets — Garden City
Leigh's Computers — New York
Mnemonics — North Port
Software Centre International — Forest Hills
Software & Such — Scotia
Software City — Forest Hills
Software Supermarket — Kenmore
Star Tech Systems — Massena
Syntaire Trading — New York
Tadiran — New York
Tedrow Business Products — Rochester
The Computer Cellar — Albany
The Working Computer — Stoney Brook
Video Computer Bin — Brooklyn
Video Computer Center — Rome
Video Connection — Schenectady
Village Computer Inc. — New York
World Computers Inc. — Hicksville

North Carolina
Digitz — Raleigh
Games 'N' Gadgets — Raleigh
Software City — Raleigh
Software City — Charlotte
TDS Computers — Durham
Triad Computers — Greensboro

North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
The Computer Store — Minot

Ohio
Computer Network — Newark
Computer Site — Cleveland
Earthrise Micro Systems — Columbus
Electronics Boutique — Columbus
Fairborn Home Computer — Fairborn
Games 'N' Gadgets — North Randall
Lakes Consumer Electronics — Akron
Main Office Supply — Coshocton
Micro Center — Columbus
Microwave Magic — Fairfield
North Coast Programming — Willoughby
Personal Computer Co. — Youngstown
Program Store — Columbus
Quality Computer Appliances — Toledo
Saxton Computer Center — Sandusky
Software Centre International — North Olmsted
Software and More — Cincinnati
Software City — Centerville
Software Connection — Mayfield Hts.

Oklahoma
Colonial Video & Computer — Bartlesville
Micro Link Computer Store — Tulsa
P C Tech — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton

Oregon
Andromeda — LaGrande
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Computron Business Systems — Portland
Software Express — Eugene

Pennsylvania
Alpha Omega EBE — Warren
Basic Computer Systems — Hermitage
Circle Computer — Ephrata
Computers and Games — Reading
Data Softique Computerware — Pittsburg
Downington Computer Center — Downington
East Coast Software — Hershey
Electronics Boutique — North Wales
Electronics Boutique — Langehorne
Electronics Boutique — Exton
Electronics Boutique — Norristown
Electronics Boutique — Pittsburgh
Electronics Boutique — King of Prussia
Electronics Boutique — Whitehall
Games 'N' Gadgets — Lancaster
Games 'N' Gadgets — King of Prussia
Games 'N' Gadgets — Media
Games 'N' Gadgets — West Mifflin
Gene's Books — King of Prussia
North Central Digital Systems — Danville
Pittsburgh Computer Store — Pittsburgh
The Electronic Boutique — Norristown
Triangle Computers — Indiana

Rhode Island
International Computer Services — Johnston
Micro Limits — Smithfield
Micro Limits — Warwick

South Carolina
Avcom International Inc. — West Columbia
Electronics Boutique — Greenville
Horizon — Greenville
Micro Computer Depot — Sumter
Watson's Jewlery — Chesterfield

South Dakota
Computer Business Center — Rapid City

Tennessee
ComputerLab of Memphis — Memphis
Games 'N' Gadgets — Memphis
Software First — Nashville

Texas
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Colonial Video — Houston
Computer Age — Houston
Computer Age — Dallas
Computer Magic — Austin
Computer Revelations — Houston
Computer Time Inc. — Marshall
FNS Computer Services — Texarkana
Metropolitan Computer — Richardson
Micro Search — Houston
Softex — Watauga
Software Library — Wichita Falls
Software Plus — El Paso
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stop — Abilene

Utah
Computers Plus — Murray

Vermont
Ray Supply Inc. — South Burlington

Virginia
Colony Ltd. — Hampton
Discovery — Falls Church
Family Computer Center — Fairfax
Games 'N' Gadgets — Hampton
Games 'N' Gadgets — Virginia Beach
Games 'N' Gadgets — Virginia Beach
Software Central — Norfolk
Software City — Richmond
Virginia Micro Systems — Woodbridge

Washington
A.P.P.L.E. CO-OP — Renton
Apple Valley Computers — E. Wenatchee
Bull Dog News — Seattle
Byte World — Kent
Com-Soft Computers — Everett
Computers + Inc. — Tacoma
Computer Outlet — Edmonds
Family Computers — Lynwood
Kitsap Amiga Operator System — Bellview
Micro Age Computer Store — Bellview
Micro Specs Inc. — Spokane
Nibble Sts — Tacoma
Omni International — Seattle
Programs Plus — Seattle
Take-A-Byte — Kennewick

West Virginia
Layhill Newstand — Charleston

Wisconsin
Celetron Computers — Racine
Computer Software Center — Milwaukee
Fox Valley Personal Computer — Oshkosh
Mom's Computers — Stevens Point
Software Shoppe — Fond Du Lac
TMW Software — Greenbay
TMW Software Inc. — Wausau

Wyoming
Byte Emporium — Laramie

Canada
Compute or Play — Alberta
Computerworks — Alberta
Computer Shop — Alberta
Data Pro — Alberta
Computer Shop of Calgary — Alberta
The Software House — Alberta
Hindsin Computing — Alberta
Red Deer Micro Systems — Alberta
Memory Land — Alberta
Computer King — British Columbia
Cent — British Columbia
Nutec — British Columbia
Sprite — British Columbia
Strider — British Columbia
Super Software — British Columbia
Canadian Computer Era — Manitoba
Micromart — Manitoba
ComputerWorld — Manitoba
J.W. Allen — Manitoba
Kobetak — Manitoba
Minicomp — Manitoba
Arkon — Ontario
Chatam Computer — Ontario
Computerland — Ontario
Compsec — Ontario
Electronics 2001 — Ontario
Hastings Data — Ontario
The Computer & You — Ontario
Thornhill — Ontario
Software Co. — Quebec
Deltronic Solare — Quebec
Microdata — Quebec
Info Mercier — Quebec
Laval Micro — Quebec
Vision Tronique — Quebec
Distribution Visipro — Quebec

Distributed in Canada by
Avita Software Distributors — Concord, Ontario
Compulit Distributors — British Columbia
Phase 4 Dist. Inc. — Calgary, Alberta

Argentina
Logic Computer — Buenos Aires

Australia
High Technology Computer Supplies — Brighton, Melborn
Maxwell Office Equipment — Abbotsford, Melburn

Japan
Pineapple 6502 — Tokyo, 111

Saudia Arabia
Abdulla Fouad & Sons — Daman

Switzerland
Software Land AG — Zuerich

Amazing Computing™ is also available in most B. Dalton Booksellers stores, B. Dalton Software stores, and Software Etc. locations.

Amazing COMPUTING™

Publisher:	*Joyce Hicks*
Circulation Manager:	*Doris Gamble*
Asst. to the Publisher:	*Robert James Hicks*
Traffic Manager:	*Robert Gamble*
Managing Editor:	*Don Hicks*
Submissions Editor:	*Ernest P. Viveiros Jr.*
Hardware Editor:	*Ernest P. Viveiros Sr.*
Amicus & Tech. Editor:	*John Foust*
Music & Sound Editor:	*Richard Rae*
Art Director:	*Keith M. Conforti*
Advertising Manager:	*John D. Fastino*
Copy Editor:	*Michael T. Cabral*
Production Manager:	*Mark Thibault*
Assistant PM	*Keven P. Desmarais*

Advertising Sales & Editorial
1-617-678-4200

Special thanks to:
Lynn Hathaway
Donna Peladeau
Traci Desmarais
Pilar Medeiros
Betsy Piper at Tech Plus.
& Paul Boden at
Software Supermarket

Amazing Computing™ (ISSN 0886-9480) is published by PIM Publications, Inc., P.O. Box 869, Fall River, MA 02722. Subscriptions: In the U.S. 12 issues for $24.00; in Canada & Mexico, $30.00; Overseas; $35.00. Printed in the U.S.A.

First Class or Air Mail rates available upon request.

PIM Publications, Inc. maintains the right to refuse any advertising.

PIM Publications, Inc. is not obligated to return unsolicited materials. All materials requesting return must be received with a Self Addressed Stamped Mailer.

Amazing COMPUTING™

Volume 2, Number 9

CONTENTS

Amazing Reviews...

Amazing Interviews...

Amazing Programming...

Amazing Columns...

Amazing Departments...

From the Editor:

Amiga and Productivity
Last month, Amazing Computing™ went to great lengths to demonstrate the variety of entertainment software available on the Commodore-Amiga. It is fair to say the Amiga is gifted with a variety of tools (sound, graphics, hardware, etc.) to produce entertaining software. When these same tools are applied to everyday tasks, the Amiga excells.

To give you an idea of the depth and variety of the current Amiga productivity tools, we spent two extra days in layout, simply making the articles fit our presses. There is simply too much material. Everyday, new Amiga software and hardware products are announced. Press releases constantly promise the moon. It is nice to see that, in some cases, the moon is shining bright.

Productivity
The programs and hardware described in this issue are just a few of the productivity tools emerging for the Amiga. From new versions of old favorites, such as Analyze!™ 2.0 and Gimoz™ 2.0, to brand new Amiga style applications such as Microfiche Filer™, the Amiga third party developer network is rapidly supplying better Amiga-oriented software.

This constant acceleration of better Amiga products demonstrates the inroads the Amiga has made in the thinking of developers. It shows a maturity of the Amiga market and an acceptance of the tools and abilities the Amiga offers.

Programming
Not to be overcome by productivity aids, we also have included two reviews of the new AC/Basic Compiler by Absoft. Regardless of the theme of an issue, we always consider programming to be one of the Amiga user's most useful tools.

The Amiga has a ways to go
Before this piece begins to sound like an Amiga pep rally, I would like to make one point. Amiga products still have a ways to go. No matter what is in the following pages, each reader will have at least one improvement to suggest for these products. This outlook in no way implies the developers have been lax in their performance; it simply means that nothing is static.

With each Amiga product introduction, developers see a new opportunity. The Amiga 500 (and soon, the Amiga 2000) will rapidly increase the installed base of Amigas. This "new territory" should draw more developers into the Amiga community. The result? Better products, with more features and better prices.

A very good friend of mine has a favorite saying, "Constant change is here to stay." As far as the Amiga is concerned, this is great news.

Fred Fish
I had hoped I would never need to address the subject of Software Piracy with Amiga owners. My belief that intellegent, farsighted users who choose the Amiga as their computer of chioice, would see the eventual folly of Software Piracy. In most cases, I am right. However.....

Piracy & Undue Harm
Unfortunately, someone in the Amiga software community must feel it helpful to remove copyright notices and publisher's names from software and place these programs in the Public Domain.

This situation is a shame, since Fred Fish saw two such programs, and felt they would be useful additions to his collection. Not knowing the software's origin, Mr. Fish released disks 80 and 88 with these programs.

One program was Inovatronics' "M," with a retail price of $59.95. Of course, Inovatronics was upset and contacted Mr. Fish. Mr. Fish immediately stopped production, but is still waiting notification of possible criminal charges.

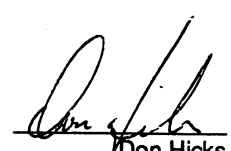
Mr. Fish retired numbers 80 and 88 from the collection to ensure that the copying process under his name comes to a halt. This unfortunate circumstance has left a hole in a very fine collection of Public Domain Software.

Although there are pros and cons of the piracy issue, one factor remains. Piracy hurts. The repercussions of an act of software piracy can go far beyond your disk drive.

The piracy included here involved removing all copyright notifications. This action goes beyond "backing up" a piece of software. No matter how you cloud the issue, theft is the bottom line. . . and the theif has left no trail. Only the innocent remain to be accused.

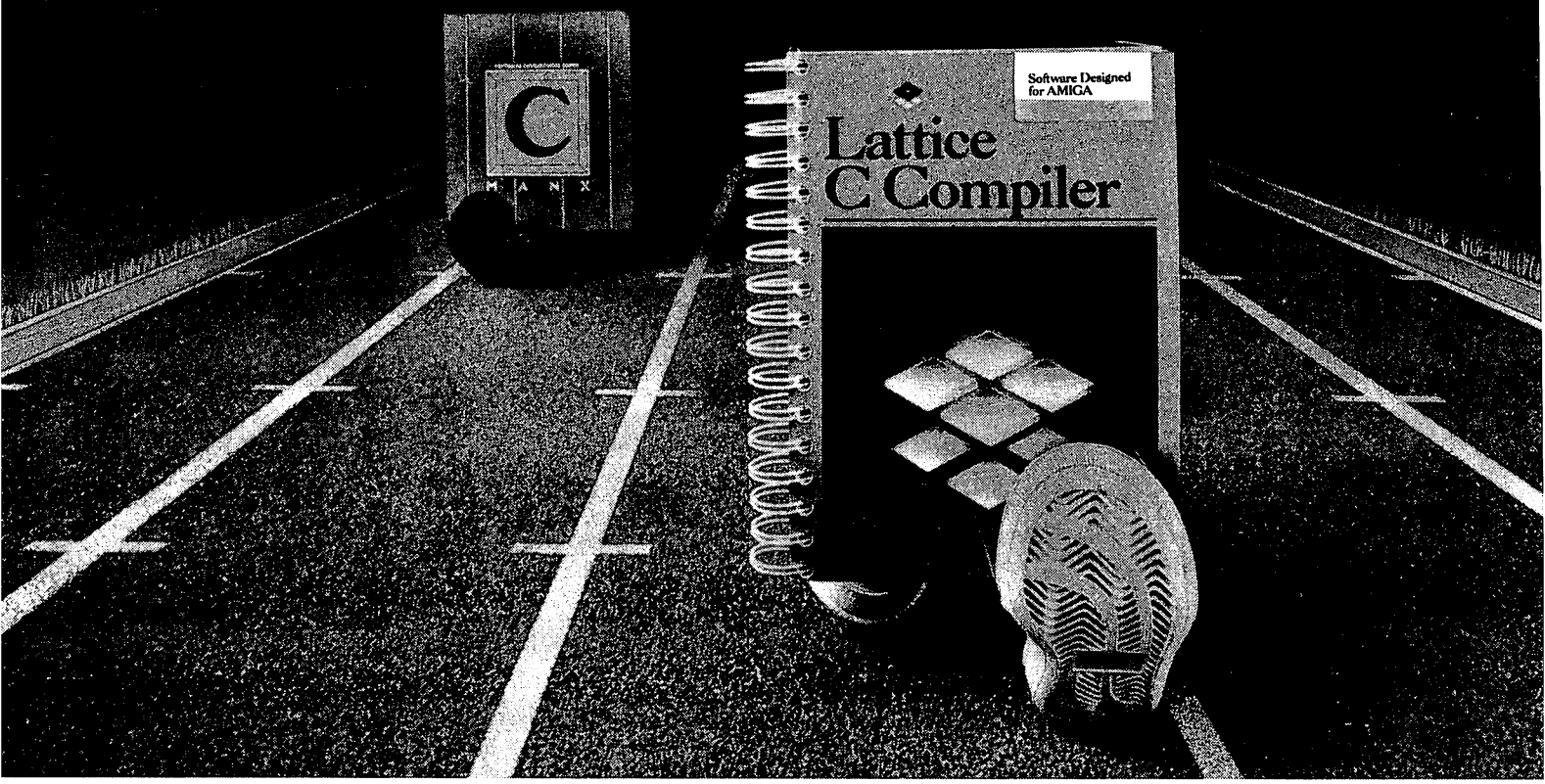
In short, the name of a respected Amiga contributor has been tarnished. The Fred Fish collection also now has two "holes" which will require explanation to new users for as long as the collection exists. Most important, a good friend and colleague faces possible criminal charges and penalties for a crime over which he had little or no control.

Software Piracy is a crime! It hurts eveyone. If this has never been clear before, there should be no doubt now.



Don Hicks
Managing Editor

C who's winning the race. Lattice C for Amiga.



Lattice C has long been recognized as the best C compiler. And now our new version 4.0 for Amiga™ increases our lead past the competition even further.

Ready, set, go. The new Lattice AmigaDOS C Compiler gives you faster, more efficient code generation and support for 16 or 32-bit integers. There's direct, in-line interface to all Amiga ROM functions with parameters passed in registers. What's more, the assembler is fully compatible with Amiga assembler syntax.

More great strides. The linker, Blink, has been significantly enhanced and provides true overlay support and interactive recovery from undefined symbols. And you'll have a faster compile and link cycle with support for pre-linking.

	Lattice® Version 4.0	Manx® Version 3.40
Dhrystone	1294 Dhrystones/second	1010 Dhrystones/second
Float	22.20 Secs. (IEEE Format) 10.16 Secs. (FFP Format)	98.85 Secs. (IEEE Format) 17.60 Secs. (FFP Format)
Savage (IEEE)	47.67 Secs./.000000318 Accuracy	119.6 Secs./.000109 Accuracy

There's no contest. Standard benchmark studies show Lattice to be the superior C language development environment. With stats like these, it's no wonder that Commodore-Amiga has selected Lattice C as the official Amiga development language.

Going the distance. You'll experience unsurpassed power and flexibility when you choose from several cost-effective development packages. There is even a full range of supporting products, including a symbolic debugger, resource editor, utilities and specialized libraries.

You'll discover that your software purchase is backed by an excellent warranty and skilled technical support staff. You'll appreciate having access to LBBS—one of the world's first 9600 baud, 24-hour bulletin board services. And you'll be able to conference with other Lattice users through the Byte Information Exchange (BIX) network.

Cross the finish line. Order your copy of the Lattice AmigaDOS C Compiler today. We'll supply the speed. You bring the running shoes.

Lattice
Subsidiary of SAS Institute Inc.

Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

Analyze! 2.0

reviewed by Kim Schaffer

An excellent way to get good business output at a reasonable cost.

I listened to the boards rant and rave about Analyze! 2.0. I tried the earlier version and was not impressed. The early version was all mouse driven and had no graphics. The boards promised great graphics and macro capability, so I thought I'd try version 2.0. This time I really tried it; I even read the manual. The manual is very good with few mistakes. It is also easily readable and definitely a decent reference book.

One of the best features of this program is that the mouse and the keyboard are integrated into one tight package. Nearly anything you can do with the mouse can be duplicated with the keyboard. If you know exactly what you want to do, the keyboard is the fastest way. If you need a little help, using the hidden windows goes a long way. The program defaults to workbench colors and memory conservation seems to be the most important task.

The program begins by allocating memory for the data, plus 16 kilobytes. The window is resizable and supports the interlace mode. The maximum window width is approximately 76 columns and supports up to 44 lines of data per screen, using the interlace mode. The worksheet commands can be entered through the keyboard by proceeding the command with "/" or by using the hidden menus. The cell location, range selection and window scrolling can also be accessed with the keyboard or mouse.

Analyze! reads and saves in either Lotus ".wks" files or its own format. It can also transform the files between these formats. Analyze! cannot, however, read the Lotus 2.0 format, ".wk1." The program loads in the data file with about 16 kilobytes of additional space (unless you manipulate the startup to do otherwise).

The program itself is approximately 230 kilobytes and leaves room for approximately 100 kilobytes of usable data space for a 512k machine. The one problem you may eventually run into is that you is cannot increase memory allocation without reloading the program. The program should try to allocate the additional memory as needed and then inform the user to reload only if necessary.

Analyze! can use up to 27 macros, including one that executes immediately after loading.

Analyze! has two very strong points in its favor: ready-made graphics and macros. The graphics are very well done and almost automated for easy use. The graphs are resizeable and fit most any horizontal to vertical ratio. A single problem stems from the fact that the minimum window sizes are not set correctly and overwrite the borders in the minimum sized windows.

The graphs can be in four color or eight color mode. The four color mode uses the least memory. The eight color is in its own custom screen with the brightest colors. The modes' operation is nearly the same, with one notable exception: the eight color mode does not respond to normal keyboard entry when viewing the custom screen.

Analyze! can use up to 27 macros, including one that executes immediately after loading. Macros are sets of commands entered as text in a single cell or a group of cells. The cells must be grouped vertically, executing from the top downward. Menus can also be created through macros, so you can customize a worksheet for any application, including voice, tone beep and graphics.

Some items deserve rewriting or inclusion in Analyze!. The file handler window is great for most things, since you don't have to mess with the mouse to load, save or delete your files or resume the program. If you want to change your file directory, you still must use the mouse. I think this quirk could have been overcome by selecting the directory first, *then* the file name.

Another point is that all negative numbers are shown in red, as long as you are using the standard workbench colors. If you don't want the negative numbers highlighted, or you want some other numbers highlighted in a different color, you are out of luck. Likewise, if you don't like the fonts used in the graphs, you are agian out of luck (unless you want to change the names around in the font directory. I would not recommend this option).

The last difficulty I found is the same for all Amiga spreadsheets I've seen. The printer setup string is always printer specific. I would like to see a standard Amiga command set used for the PRT: port. There is more to be said for making the print commands standard for the computer than for the printer. I would also like to see the option preferences included in the setup.

There are some commands that I think are missing from Analyze!. The first is a function menu. The functions are the only major items not supported by the mouse. I see the mouse as a kind of help and I think it would work nicely with a function menu. A "fast" menu that would let you build a formula by using just the mouse would greatly enhance the function utility.

Another command that is lacking is a search (or find) command that lets you find the area you are interested in, without having to scroll the entire file. An extract command is also necessary to allow you to pull the particular subgroups from the main block.

The final item I would like to see is a map function indicating which locations are being used. For example, when using a spreadsheet for various calculations and grouping the data together in one table, you might use several different areas to do the different calculations.

The resizing feature of the worksheet and the graphs is a fantastic! The programmers really tried to make this program fit into what ever space you have available. In fact, as I write this article, I am using Scribble! and Analyze! together on an interlaced screen (yes I do have extra memory). These two programs also interact to help you create reports with tables from Analyze!, or use a table from your report to perform some analysis in Analyze!. The transfer is done with the cut, copy and paste from Scribble! and the range import and export function of Analyze!.

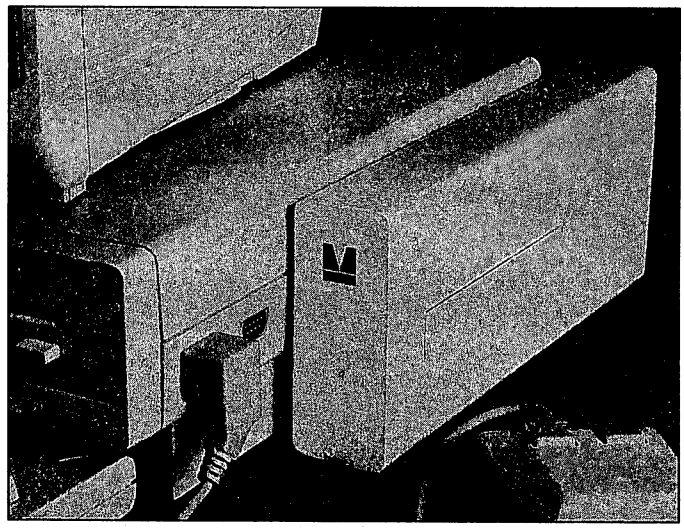
Another program by Micro-Systems Software, Flipside! enables you to print your output sideways. These three programs offer an excellent way to get good business output at a reasonable cost and are soon to be marketed under one title, The Works!

•AC•

AVAILABLE NOW!

StarBoard2

Auto-Configuring Fast RAM
Zero Wait States
User Expandable from 512k to 2 Megabytes
Bus Pass-Through
MultiFunction Option: battery/ clock, FPU, parity, Sticky-Disk



If you've owned your Amiga® for a while now, you *know* you definitely need more than 512k of memory. You probably need *at least* double that amount...but you might need as much as an additional two megabytes. We want to urge you to use **StarBoard2** as the solution to your memory expansion problem –and to some of your other Amiga-expansion needs as well!

It's small, but it's BIG–

Since most of you want to expand your Amiga's memory without having to also expand your computer table, we designed **StarBoard2** and its two optional "daughterboards" to fit into a sleek, unobtrusive Amiga-styled case that snugly fastens to your computer with two precision-machined jackscrews.

The sculpted steel case of **StarBoard2** measures only 1.6" wide by 4.3" high by 10.2" long. You can access the inside of the case by removing just two small screws on the bottom and pulling it apart. We make **StarBoard2** easy to get into so that you or your dealer can expand it by installing up to one megabyte of RAM on the standard **StarBoard2** or up to two megabytes by adding in an Upper Deck.

This card has decks!

The basic **StarBoard2** starts out as a one megabyte memory space with 0k, 512k, or one megabyte installed. If you add in an optional **Upper Deck** (which plugs onto the Main Board inside the case) you bring **StarBoard2** up to its full two megabyte potential. You can buy your **StarBoard2** with the Upper Deck (populated or unpopulated) or buy the Upper Deck later as your need for memory grows.

And you can add other functions to **StarBoard2** by plugging in its second optional deck –the Multifunction Module!

StarBoard2: functions five!

If we count Fast Memory as one function, the addition of the **MultiFunction Module** brings the total up to five!

THE CLOCK FUNCTION:

Whenever you boot your Amiga you have to tell it what time it is! Add a MultiFunction Module to your **StarBoard2** and you can hand that tedious task to the battery-backed, real-time clock/calendar. A small piece of MicroBotics software in your WorkBench Startup-Sequence reads the clock and automatically sets the time and date in your Amiga. And the battery *is* included (we designed it to use an inexpensive, standard AAA battery which will last at least two years before needing replacement).

THE FLOATING POINT FUNCTION:

If any one aspect most characterizes the Amiga it's *fast* graphics! Most graphic routines make heavy use of the Amiga Floating Point Library. Replacing this library with the one we give you with your MultiFunction Module and installing a separately purchased Motorola 68881 FPU chip in the socket provided by the Module will speed up these math operations from 5 to 40 times! And if you write your own software, you can directly address this chip for increased speed in integer arithmetic operations in addition to floating point math.

THE PARITY CHECKING FUNCTION:

If you install an additional ninth RAM chip for every eight in your **StarBoard2**, then you can enable *parity checking*. Parity checking will alert you (with a bus-error message) in the event of any data corruption in **StarBoard2**'s memory space. So what good is it to know that your data's messed up if the hardware can't fix it for you? It will warn you against saving that data to disk and possibly destroying your database or your massive spreadsheet. The more memory you have in your system the more likely it is, statistically, that random errors will occur. Parity checking gives you some protection from this threat to your data residing in Fast RAM. Note that the Amiga's "chip" RAM cannot be parity checked.

THE IMMORTAL MEMORY DISK FUNCTION (STICKY-DISK):

When you've got a lot of RAM, you can make nice big RAM-Disks and speed up your Amiga's operations a lot! But there's one bad thing about RAM-Disks: they go away when you re-boot your machine. Sticky-Disk solves that problem for you. It turns all of the memory space inside a single **StarBoard2** into a Memory Disk that will survive a warm-reboot! When your Amiga attempts to grab a **StarBoard2** in Sticky-Disk mode, a hardware signal prevents the system from acquiring the **StarBoard2** as FastRAM (and thereby erasing your files) –instead it is re-recognized as a Memory Disk and its contents are preserved intact. If you want to work rapidly with large files of data that are being constantly updated (such as when developing software) you can appreciate the Sticky-Disk!

Fast RAM –no waiting!

StarBoard2 is a *totally* engineered product. It is a ZERO WAIT-STATE design, auto-configuring under AmigaDOS 1.2 as Fast RAM. Since AmigaDOS 1.1 doesn't support autoconfiguration, we also give you the software to configure memory in 1.1.

Any applications software which "looks" for Fast RAM will "find" **StarBoard2.** And you'll find that your applications run more efficiently due to **StarBoard2** on the bus.

A passing bus? Indeed!

What good is an Expansion Bus if it hits a dead end, as with some memory cards? Not much, we think –that's why we carefully and compatibly passed through the bus so you could attach other devices onto your Amiga (including another **StarBoard2**, of course!).

The sum of the parts...

A really nice feature of the **StarBoard2** system is that you can buy exactly what you need now without closing off your options for future exapansion. You can even buy a 0k **StarBoard2** (with a one megabyte capacity) and populate it with your own RAM (commonly available 256k by 1 by 150ns memory chips). When you add **StarBoard2** to your Amiga you have a powerful hardware combination, superior to any single-user micro on the market. See your Authorized Amiga Dealer today and ask for **StarBoard2**

SUGGESTED RETAIL PRICING:	
StarBoard2, 0k (1 meg space):	$349
StarBoard2, 0k (2 meg space):	$395
StarBoard2, 512k (1 meg space):	$495
StarBoard2, 1 meg (1 meg space)	$595
StarBoard2, 2 megs installed:	$879
StarBoard2, 2 megs & MultiFunction:	$959
Upper Deck, 0k (1 meg space):	$ 99
MultiFunction Module:	$ 99
also available:	
Standard 256k memory card:	$129
MAS-Drive20, 20 meg harddisk:	$1495
MouseTime, mouseport clock:	$ 50

MicroBotics, Inc.
811 Alpha Drive, Suite 335, Richardson, Texas 75081 / (214) 437-5330

Impact Business Graphics

from Aegis Development

by Chuck Raudonis
Plink: CWR

If the Amiga is to be accepted in the corporate world of Big Blue and Scottish Apples, the machine will have to have a solid base of useful, friendly software developed for it. With the advent of Impact from Aegis, the first salvo in the business war has been fired. Impact is a fully functional, comprehensive presentation quality graphics generation system.

Impact is a unique package. It is unlike traditional graphics packages on other computers. Impact combines four tools in one package. The basic tool is the SLIDE BUILDER.The slide builder is the "canvas" upon which the final product is built. The other three tools are used to build pieces of the final product. Each tool is specialized to edit a specific type of graph piece.

The GRAPH BUILDER is the tool that builds the basic graphic image and can create Bar, Line, Area and Pie charts. There are Horizontal, Vertical, and 3D styles for each graph. In all, there are 28 different types of graphs that the GRAPH BUILDER can generate. The TABLE BUILDER tool allows the user to build and manipulate blocks of text.The ICON BUILDER allows the user to create small, detailed graphic images that can be used as custom brushes. The image can be "rubber stamped" onto the slide by moving the cursor and touching the select button. The ICON BUILDER also has an additional function when used in conjunction with the GRAPH BUILDER, this function will be covered later in this article.

The difference between Impact and other business graphics packages is Impact's role as an object oriented editor. Packages such as Images and Deluxe Paint are pixel oriented editors. Once a shape is drawn in a pixel oriented editor, it becomes part of the image. Once it is drawn, "UNDO" button not withstanding, it is a permanent part of the image. In an object oriented editor, each shape is retained as an independent, identifiable, shape that can be manipulated independent of the rest of the image.

Lets look at the individual tools....

The TABLE BUILDER allows the user to edit a block of text for inclusion into the final slide. Both the SLIDE BUILDER and the TABLE BUILDER have text manipulation capabilities. The text created with the TABLE BUILDER will be treated as a single object when it is included in the slide. Text created with the SLIDE BUILDER will be treated as individual objects. Each has its place in the creation of a graph. It is nice to have both options available.

Both tools have a unique capability in text editing. The fonts that are available can be resized from a minimum of 4 point to 18. This flexibility allows the user to choose the text size that is needed for the application. In addition to changing the size of the text, the user can modify the text by adding bold, italics, underline and shadow. The package comes with five different fonts.

Impact combines four tools in one unique package

This flexibility does not come free. Impact does not support additional fonts. This limitation could be due to the need for flexibility of sizing the fonts or it could be a design feature. Images does not support external custom fonts either, so this design might be a specification from Aegis. It would be nice if Impact could make use of the fonts available in the public domain or those that are created by the font editor.

The GRAPH BUILDER tool is supported by a spreadsheet-like series editor. This tool allows the user to create the series of numbers that are used to create the graphs. There are no calculation capabilities in the series editor. This editor simply provides a convenient way of organizing the data needed for the GRAPH BUILDER. The GRAPH BUILDER can plot only up to eight series of data. This limitation can present a problem in some business applications. Once the set of series has been defined, the type of graph is selected. Once this is accomplished, the graph is drawn. The user also has the ability to override the scale and axis orientation.

The second use for the ICON BUILDER is to create an icon graph. An icon graph is a bar graph that uses the selected icon in place of the standard bars. For example, if one wanted to make a graph outlining the sales figures for a film company, an icon could be created that looks like a box of film. When the graph is generated, the bars would be stacks of film boxes. This option is for special use, but it is a nice effect when it is used sparingly.

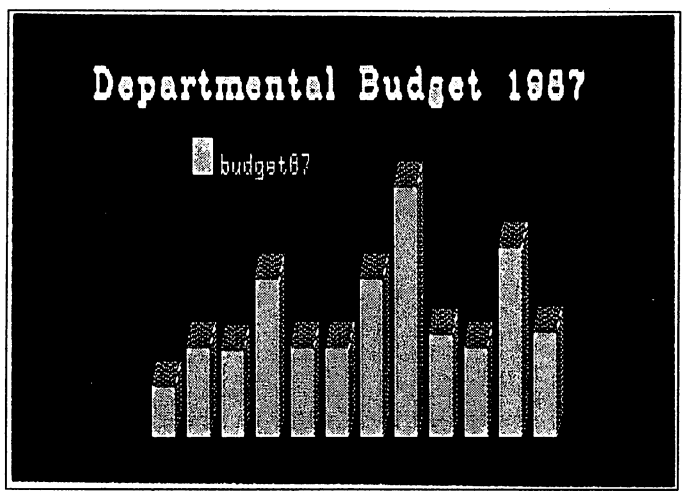
The ICON BUILDER is a mini-paint program that will define and edit the icons for the GRAPH BUILDER or for use as a custom brush. The ICON BUILDER has the capability of fill, freehand drawing, arcs and lines. The icon can also be rotated and shifted in the window.

Once the graph is built using the GRAPH BUILDER, it is copied into the SLIDE BUILDER. It is then treated as a single object. The same applies to a block of text created with the TABLE BUILDER or an icon created with the ICON BUILDER. Once in the SLIDE BUILDER, the objects can be manipulated, resized and augmented. Multiple graphs can be placed on one slide or a slide can be all text, if that is what is needed. One of the nice features of the SLIDE BUILDER is the "undo" function. Because IMPACT is an object oriented system, the undo is much more powerful than the undo in other graphics packages. The undo is an intelligent undo. Every action that was applied to the slide can be backed out --- the slide can be backed down to a blank screen.

Impact includes a separate program that functions as a slide show director. It takes the slides that were created with the SLIDE BUILDER and programs a presentation to the screen. The slides will appear in the selected order and the program will make the slides appear in various ways. The slides can fade in or out, wipe in, spiral in, or fade in by a random dithering pattern.

Impact has the ability to import data from external programs into the series builder of the GRAPH BUILDER tool. This idea is good in concept, but the format that the package requires is so rigid, that unless a package is specifically written to feed Impact, it will not be very useful.

The manual is very well written. It is aimed at an Amiga novice, assuming the reader knows nothing about the computer. It walks the user through the use of the system starting with the mouse and gadgets. Considering that the package is aimed at the business user, the level of Amiga sophistication will not be high in prospective customers. A well-written manual will make a first time user much more comfortable.



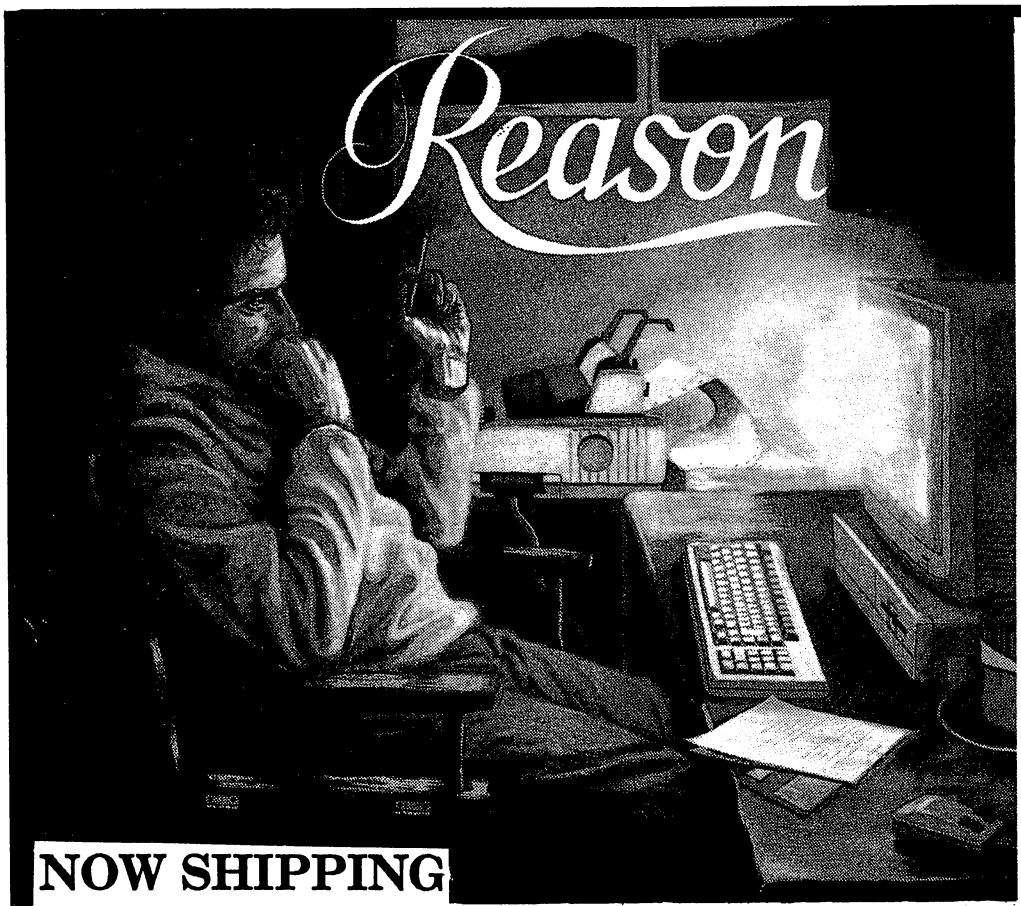
Impact is not without its faults. The program is slow. If a slide has many object on it, it takes a long time to reconstruct, if it is covered by another window or one of the other tools. Considering the intended business target market for this product, the lack of speed will be a problem with users of the program.

In addition, in an attempt to make the program self documenting, all features are available from the pull down menus. This touch is very nice for the new user. For the experienced user, pull downs are a nuisance. To have to go back to the menu for every option is a very tedious affair. It would have been nice if Aegis would have added some "Amiga" Key" combinations for the key functions. That way, an inexperienced user could use the menus, but an experienced user would have the option of using the key combinations to activate the desired functions.

Impact does have the ability to save the graph as an IFF file. Unfortunately, the graph is saved with a border of black. When the medium resolution image is loaded into a paint package, the border must be filled in manually.

The last problem with the package is the price. The package lists for $199. What happened to the concept of good software for under $100??? If Aegis would take the package back to the drawing board, rewrite it into Assembly to get the speed up a bit and add some Amiga key combinations or function keys, I think it would be a better product. It is a very useful product in its current state, but I think it would be a much more widely accepted program in a slightly revised form.

•AC•



NOW SHIPPING

Special

Introductry Price

Complete Communications Package

300/1200 1 Year warranty

300/1200 Fully Hayes compatable Modem - 2 Year warranty

$129.00
(Modem, Cable & Software)

300/1200/2400 Fully Hayes compatible modem CCITT - 2 Year warranty

$249.00
(Modem, Cable & Software)

Call or write for information about our other great products for the Amiga or our Demo disk $5.00

The REASON system is a series of programs designed to aid writers and editors in editing documents.
REASON programs do three things:

*proofread input text *analyze the style of input text *provide help about English usage

Many options give editorial comments and suggestions.

The **REASON** system finds potential errors, then you decide which potential errors need correcting. Thoughtful use of the **REASON** system can help both the experienced and inexperienced writer.

With the REASON **system, there are six main options:**

1. **Prose** describes the writing style of a document, namely, readability and sentence characteristics, and suggests improvements.
   Prose compares a document with standards for one of several document types. INSTRUCTIONAL TEXT will compare input text with good training documents. TECHNICAL MEMORANDA will compare input text with good technical memoranda. And USE CUSTOM STANDARDS will compare input text with any user created standard.
2. **Style** finds sentences that contain passive verbs, expletives, noun nominalizations, and multiple nominalizations. Also, Style will give a readability level for each sentence in the input text or find sentences that are equal to or greater than a specifically defined readability level. Another function performed by style is to find sentences that have a specifically defined length (number of words contained in a sentence).
3. **Word Analysis** will check the input text for general diction, sexist terms, sentences that contain forms of the verb "to be", acronyms and abstract words.
4. **General Structure** checks input text for general organization, general topics, sentence breakdown (parts of speech) and syllable breakdown (syllable count of each word)
5. **Proofread Document** checks for possible spelling errors, double words, possible punctuation errors, diction and split infinitives.
6. **Extra** allows access to AMIGA Preferences and Build Custom Prose Standard.

Requires an AMIGA computer with 512K
AMIGA is a Registered Trademark of Commodore Amiga

$395.00

THE OTHER GUYS

55 North Main Street
Suite 301-D
PO Box H
Logan Utah 84321

(801) 753-7620
(800) 942-9402

VISA MasterCard

Microfiche Filer

A database/data-storage and retrieval system unlike any other

by Harv Laser
People Link: CBM*HARV

When the topic of database programs is raised, the most commonly asked question is "Why do I need one?" After a discussion about address books, inventories, cataloging a video tape collection, mailing lists or even kitchen recipes, the curious user has some idea of the usefulness of such a program. The next questions are usually "Well, which is the best database for me?" or "I've heard they're really difficult to use. Are there any easy ones out there?"

I will assume you have decided you need a database program or that you are shopping around because you already own one, but are not happy with it. Maybe you bought one based on a friend's recommendation, but after bringing it home and struggling with it for a while, you decided that it was too difficult to use, not powerful enough, more powerful than you need, doesn't take advantage of your Amiga's capabilities or simply not enjoyable to use. (For the ultimate in user-hostile databases, take a look at what our friends with IBM compatibles face when they load up DBase II: a "." dot prompt!).

Maybe your address or mailing list has simply grown to the point where you need something more powerful than a simple BASIC sorting and printing program. Perhaps you've collected an enormous number of IFF pictures and when friends come over, you want to impress them with some new pictures. You can't remember what you've named those pictures, so you end up displaying dozens of them before you find the right ones.

Well, these scenarios go on forever. There are about as many different database programs as there are ways to use them!

An Amiga owner has an advantage over his PC friends when it comes to selecting a database. He might not have as many to choose from, but the sheer power of the Amiga's Intuition interface, multitasking and great ease of use makes the potential for databases designed especially for the Amiga simply enormous!

Microfiche Filer from ***Software Visions*** is such a program. It is a database/data-storage and retrieval system unlike any other. Based on the assumption that it is easier to find the things you need when you can see them, programmer Gary Samad has written a program with an interface that is at once elegant, simple, and powerful.

Let us take a minute and relate this new database to a real-world situation. It may be easier to grasp its differences this way.

When was the last time you went to your local public or university library to look up a book or article you needed for a research project, a term paper or something you wanted to read for pleasure? First, you went to the card file. Maybe you already had an author's name in mind, so you went to the drawers containing the card listing of the library's books alphabetized by their author. But wait, if you didn't know the author, but you thought you knew what the title might be... off to another set of drawers to search on alphabetized titles. Suppose you just went in "cold" not knowing either the author or title, but you knew your subject! Hmmm, where are those the subject drawers?

Card files are no fun. They are tedious to use, and besides, they're very low tech! A lot of microcomputer database programs use the analogy of the card file as a design theory. The advent of microfilm was a great step forward, but it was still sequential. To get from point A to point F on the film, you still had to wind through B, C, D, and E.

Enter the microfiche reader. Microfiche is a flat-sheet photographic reduction of card files (or newspapers, periodicals, etc). Its advantage over cards or film is that it can be randomly accessed. Just open a small binder containing slips of film, slide one into the viewer, move a pointer around the surface of the fiche and look at the screen.

Microfiche readers are much faster to use for searches. You know that this piece of fiche contains all the book titles (or authors or subjects) in the library from A to B, the next one C to D, and so on. It's a simple matter to grab the right piece of fiche and search through it on the fiche reader's screen to find what you want quickly. Lugging drawers full of cards to a table and flip through them. Some fancy microfiche readers even have interfaces to copying machines. Just hit a button and zap what's on the screen to the copier! Microfiche provides an instant reference guide you can use as you hunt the library's.

Computerized microfiche

The library's card file is a database. It's much easier to search for something on fiche, rather than digging through thousands of paper cards or winding through reels of film, Fiche displays data for you, spread out like a map. You simply move a "magnifying glass" across it and view portions. You can quickly fly from one area to another.

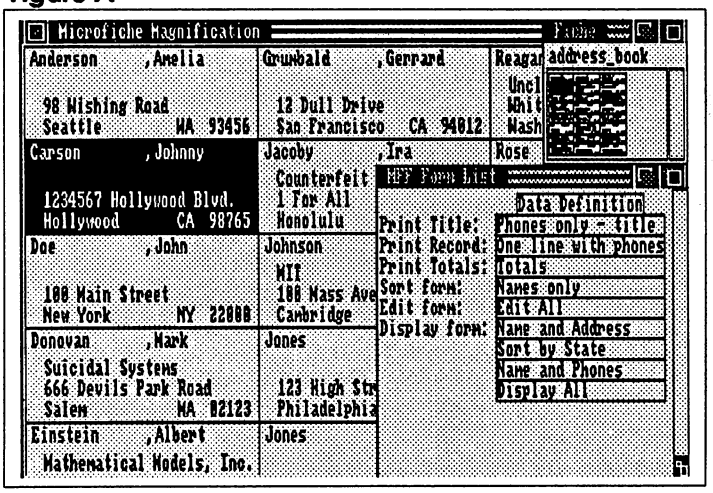
Microfiche Filer brings this capability to the Amiga computer. Rather than digging down through layers of computerized "files, folders, and cards" to search or display your data, the microfiche concept is used.

Microfiche Filer is a "graphic-oriented" database program that fully exploits the Amiga's Intuition interface. Each database you create is displayed on your monitor utilizing three Intuition windows. (See Figure A).

The largest window (when the database is active) is the "Microfiche Magnification" area. Here, all your records are displayed, neatly sorted similar to the screen of a microfiche reader. A small window in the upper right hand portion of your screen is the "Fiche sheet" (a highlighted rectangle), representing your entire database and a magnifying glass. Using your left mouse button to "grab" this magnifying "glass" and move it around your fiche. As you do, the portion of the fiche under the glass is displayed in the large magnification window.

continued...

Figure A



Don't worry about slow screen updates with Microfiche Filer-- there aren't any when you are manipulating the glass. You can grab the glass and move and wiggle it as fast as you like and the large window's contents will always keep up with your hand motions. Sweep across the fiche in any direction and instantly locate the data you're looking for! If your database is too large to fit in the fiche window (which is a reduced-pixel representation of your data), a slider gadget appears on the side of the window. This gadget lets you move to different areas of your database, as though you were changing sheets of fiche in a mechanical reader.

Suppose you want to change some data. Maybe you're using your address book database and one of your friends has recently moved. You need to erase his old address and phone number and substitute his new information.

Grab your magnifying glass, scan the fiche sheet while watching the large window. There's his name! Move your mouse to the large window and double-click on the box containing his record. A new window opens, the "Record Editor" with his individual data displayed, each field in its own smaller box. Via menu options, the cursor keys and the mouse, you can swiftly move through each record, changing as needed. Close the editor window with its gadget. Microfiche Filer will ask you if you really want to make this change. Click the appropriate gadget and the deed is done. The record is replaced back into the fiche and instantly re-sorted.

Form oriented

Microfiche Filer is a "form-oriented" database. The third default window which opens when you run the program is the "Microfiche Filer Form List" window. This window provides you with an easily-understood visual display of how your database is defined-- how you've chosen to sort it, print it, store it, edit it, and display it in the magnification window. Changing these parameters couldn't be easier, unless your Amiga grew hands and moved the mouse for you! Look at the example A picture again. Our Address book database is displayed in the "name and address" mode only. Don't like that? Want to see more? Just grab the field called "Display All" with your mouse and drop it on top of "Name and Address." The magnification window instantly reflects this change and displays the complete contents of every record, all the fields you've defined for each person's data!

At any time, you can pick up a form definition, "drop" it into one of the "slots" and your displayed data will instantly change! Microfiche Filer stores your database in RAM memory as you work with it, so these changes are virtually instantaneous. Creating new form definitions or editing existing ones is just as easy.

Memory requirements

A memory requirement warning: although Microfiche Filer is designed to work with a one-disk-drive and a 512K Amiga as the minimum suggested hardware setup, the addition of more RAM memory is highly recommended! I experimented with memory while working with Microfiche Filer by shutting off my 2 meg RAM expansion board with a little public domain program called "RAM ON\OFF".

Although Microfiche Filer still loaded the large public domain software database supplied on the program disk, the screen would only display the database up through the letter B in the alphabetical file listing, (even though the entire database was indeed loaded into memory.) Intuition windows eat memory. If you have only 512K, you'll probably have to close some Workbench windows or end other tasks. Microfiche Filer is still completely usable, but more RAM will make it much easier to use. As soon as you can comfortably afford to, the first hardware purchases you should make to add to your Amiga are a second disk drive and more RAM. You'll never regret it!

Microfiche Filer includes built in memory-management routines to prevent you from losing your data due to insufficient RAM. You'll never see the GURU due to lack of available memory. Instead, an "Emergency Save and Exit" feature will let you escape from Microfiche Filer and save your database back to disk, if you run extremely low on memory.

You can easily create forms into which you input and store your records. Since Microfiche Filer comes with a number of very handy database examples on the program disk, even if you're in a hurry and don't want to design forms from scratch, you can use one of the existing databases as a "template." Software Visions has been particularly "visionary" in their approach to supplying USEFUL examples.

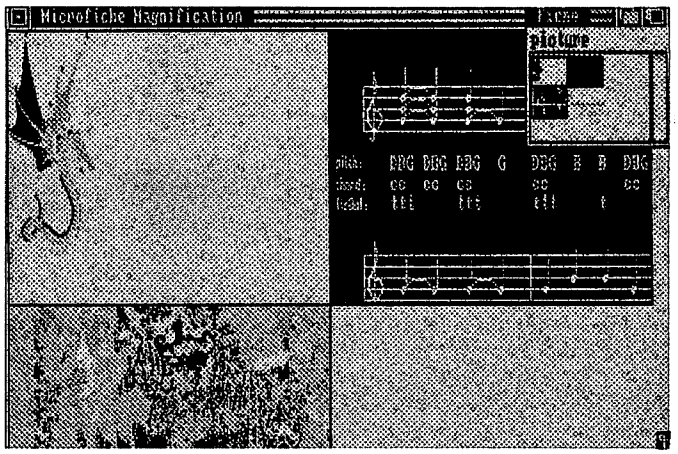
Sample databases

The sample databases include catalogs of all current Fred Fish and AMICUS public domain software disks, (which will be updated as those collections expand), some excellent "modern art" in the form of IFF pictures collected from M.I.T. artists, a commodities index of world wide currencies including their U.S. dollar equivalents and trading markets, a small database of Amiga workbench icons and a sample address & phone directory. These "free" databases (included with the purchase price of Microfiche Filer) are a pleasant bonus. Not only are they used during the "Quick Tour" portion of Microfiche Filer's excellent manual (which, like all good manuals, is ring-bound to lay flat on your desk, cross-indexed, and includes boldface and colored text to highlight important sections, features, and topics), but they provide an instant base which you can use to expand add your own data.

A Microfiche Filer database is also extremely easy to re-define at any time. This point is one of the major downfalls of most database software. Once you have set up your fields, records, and files, you might be locked into them. If you need to add a new field to every record (suppose you forgot to add a "Country" field in the address book and, after entering hundreds of records, you realize that you have some foreign addresses and no place in which to input the country name!), other software might require you to start from scratch or use some long-winded accessory program to reformat all your records. Microfiche Filer lets you add a field to your "form" at any time. There is no need to 'convert' your database.

Picture databases are handled by Microfiche Filer like no other Amiga database I've seen (Refer to Example B illustration). Each picture in your disk database, once properly cataloged and filed with Microfiche Filer, is displayed onscreen in the magnification window in a "squeezed" version of the full-sized picture.

Figure B



Again, by grabbing your magnifying glass, you can very rapidly search through your database of pictures. If you choose to view a picture with full color resolution and full screen size, just double-click on the representation of the picture, pull down a menu selection (or use the keyboard shortcut) and the picture will be displayed. Another mouse-click returns you to your Microfiche Filer screen. Microfiche Filer also has the capability to display IFF "brushes" (rather than just full size pictures), although at this time, it will not handle Hold and Modify (HAM) mode Amiga pictures. This capability will be added to Microfiche Filer in a planned upgraded.

Since Microfiche Filer uses a standard four-color Workbench screen for its display, instead of a custom screen (and in doing so, you can resize and re-stack Microfiche Filer's default windows any way you like, along with your usual Workbench windows, since everything is on the same screen), your 32 color IFF pictures will only be displayed on the fiche screen in 4-colors. (You do, of course, see the actual 32-color picture when you choose to view it full-size from the database record).

Microfiche Filer uses a special "color squeezing" algorithm to translate a 32-color picture to four colors for fiche display purposes. A color editor window is provided for you to change how those four colors are shown-- a nice feature that certainly isn't a requirement, but indicates very thoughtful programming!

More features

Microfiche Filer has "instant sorting." Again, look at the illustrated example. Dropping a sort definition in the slot next to "Sort Form" will immediately re-sort your database, in any method you like. You may sort on one or all fields simultaneously, and your data is ALWAYS kept and displayed in sorted order. There's no need for you to manually build complex "indexes," as with some other software.

Selecting records for viewing or printing is simple and quick. Grab your mouse, point at the record you want to magnify and double click. (Caution: some public domain software which modifies the mouse pointer's action may cause Microfiche Filer to act strangely. I discovered, while using Microfiche Filer, that by having a program called "Autopoint" loaded into memory, Microfiche Filer did not respond correctly to double-clicks. This quirk was not Microfiche Filer's fault, as the filer assumes you are using the normal, unmodified mouse-click and position routines built into Intuition. So, if you also have similar problems such, reboot to remove those mouse-modifying programs). Microfiche Filer supports Intuition's "Extended

continued...

Select" feature by holding down a shift key while clicking on individual records. This feature enables you to select a specific group of records. As with any powerful database software, you can qualify your selections based on the contents of individual fields.

As exemplified in Microfiche Filer's sample Addresses database, by clicking on the "Selection Editor" (or using the keyboard equivalent - Microfiche Filer supplies keyboard equivalents for all important mouse and menu functions - you are NOT mouse-bound with Microfiche Filer), you could, for example, select all the people in your database who live in Massachusetts, and then further narrow your choices by selecting those who have a home phone number listed, but no work number, and whose names fall within the A - M range. It is powerful and easy to use.

There are many printing options included with Microfiche Filer. If you have a color printer and choose to print IFF color pictures, you'll get color printouts, of course. The manual describes, in great detail, the ways in which you can vary your printer output to get just the results you desire. Microfiche Filer has an "abort printing" gadget as well (a feature commonly lacking in too much Amiga software.) If you change your mind and want to stop printing, just click "abort."

Besides using your Workbench Preferences selection, Microfiche Filer also has its own built-in Preferences window. You can customize Microfiche Filer to your own way of working. The Microfiche Filer preferences can be used for your current project or saved to disk, your choice. If you don't like Workbench ".info" clutter (saving an icon for each database), you can specify not to create icons in Preferences. There are many more possible options .

Summary: Excellent

When I set out to write this overview of Microfiche Filer, I considered different approaches. One thing I wanted to relay to you is that the power of the Amiga is finally being tapped by some extremely talented developers. Microfiche Filer is an exemplary piece of software which is a joy to use. Rather than give you a blow-by-blow description of every feature of the program, I wanted to cover some of its highlights and the innovative and important differences from other database software you might have seen.

I did not want to create a review which would serve as a substitute for Microfiche Filer's manual. Microfiche Filer comes to you on an unprotected disk. There is no hard-sector disk protection, no stick-a-dongle-in-the-mouse-port, no "look up a word in the manual" routine. This move is a gutsy one on Software Vision's part. They've created an extremely well-written, well-debugged and well-documented program and the are respecting and trusting their customers by not copy protecting Microfiche Filer in any way. This approach deserves to be applauded.

In conversations with Gary Samad, Microfiche Filer's author, I was told that Microfiche Filer had undergone "thousands of man-hours of testing" before it was released. I believe him. This Amiga program is as solid an as I have ever used. I, literally, could not make it crash my Amiga. I never saw the dreaded GURU during weeks of using Microfiche Filer. Nor did I find anything that I would consider to be a fatal "bug" that would keep me from using Microfiche Filer, or cause me to be dissatisfied.

No piece of software is "perfect"; no piece of software is every really "finished" either. Someone can always think of new features to add or ways to improve a program, but Samad has created something here that seems truly rare these days, value for your money!

I questioned Samad about future upgrades, user support after purchase, and the costs to the buyer. One thing I found lacking in Microfiche Filer, which I considered the program's only major drawback, was that it has no facility to import modified ASCII text and turn it into a database. Samad told me that many people who have seen the program have made this same comment, and he is working on allowing importing of data into a future version, as well as giving Microfiche Filer the capability to show HAM mode picture files.

Samad told me that if any bugs are discovered in his program, they will be fixed and upgrades to correct bugs will be free, or there will be a very nominal charge for a disk swap.

Telephone support (a long distance phone call if you live outside Software Vision's local phone number radius) is included in the purchase price. On the few occasions I phoned Software Visions, the call was answered before the second ring!

Microfiche Filer's manual is excellent. It doesn't waste paper describing how to turn on your Amiga or load a disk, but rather refers you to the documentation supplied by Commodore which came with your machine. It's professional and well indexed, with a quick reference guide and keyboard shortcuts. It's also just the right size for desktop use.

Lastly, for "spec sheet fans" Microfiche Filer features you'll appreciate include: Unlimited number of fields per record, all fields are variable length, virtually unlimited field length. The number of records is limited only by available memory. (512K allows 200-400 address book records; 1.5 megabytes allows 2000-3000 records. Microfiche Filer can use a full 8 megabytes of memory.)

I highly recommend Microfiche Filer if you're looking for an Amiga database program.

•AC•

Microfiche Filer Software $99.00
Not Copy Protected
Requires AmigaDOS 1.2, 512K

Software Visions Inc.
26 Forest Road
Framingham, MA 01701
Order Line (800) 527-7014

Pagesetter

a page layout program for the Amiga

by Rick Wirch

A page layout program can be used to make newsletters, handouts and even magazines. Such programs require a printer to produce output. In general, the best layout programs can output to laser printers and photo typesetting machines. Many others offer output to dot matrix printers. The term "desktop publishing" was coined for layout programs because they bring typesetting abilities to your desktop.

PageSetter, like its Macintosh rival PageMaker, was the first page layout program available for its machine. However, I do not think that PageSetter will revolutionize the Amiga as PageMaker did for the Mac. PageMaker is the program that legitimized the Macintosh for business use. It provided a new use for computers and the Mac was the only computer, at the time, that fulfilled the needs of desktop publishing. The Amiga also fulfills these needs and, since its introduction (over a year ago), page layout programs have become available for the Amiga. In this review, I compare Amiga PageSetter with Macintosh PageMaker.

My first experience with PageSetter was very pleasant. Without consulting the manual, I was able to create text, add graphics, adjust the fonts and layout and print the page. I also tried composing the same page with PageMaker. This task was very difficult, even though I am familiar with the Mac. I was forced to ask a knowledgeable co-worker for help three times. I finally made a similar layout, but I was rather mystified by PageMaker's approach.

The difference can be easily explained. PageSetter uses the same metaphors for page layout as the Amiga Intuition interface. Since I am familiar with the Amiga desktop, I immediately understood PageSetter's method of page layout. PageSetter uses the concept of boxes, just like windows on the Amiga desktop. These boxes, containing text or graphics, are resized using the lower right corner and are dragged using the upper portion of the box. You simply drag out a box, position it on the page and drop text or graphics into it.

PageMaker uses a 'column' metaphor. You specify the number of columns and width of a page. PageMaker then places columns on the page that run the length of the page and can be resized only in width. When text is placed in the column, a 'tab' appears at the start and end of the text. These tabs act like the cord on a window shade, pulling the text up and down the column. Unfortunately, this method is not powerful enough to fully express a typical page layout. Laying out a typical page (a headline across the entire top of the page and three columns of text beneath) breaks the column metaphor. The headline is not within any of the columns, while the body of text is.

Actually, any graphic image or headline text generally does not lie within the columns, while the body of a document must——a small point, but consistency always helps the user. Consistency is the philosophy that explains the superiority of icon-based operating systems.

Once something is in a box in PageSetter, you have many presentation options for the box. Boxes can have six different line styles surrounding them and can choose which sides to have lines on. The boxes, which also have many levels of gray as a background, can be transparent (see through another box), can have dropped shadows, adjustable margins and their own fonts and point sizes for text. Adjustable spacing between lines and four ways to justify complete the capabilities of the boxes. The justification also allows you to specify the amount of spacing to put between characters and words. PageSetter allows you to save all these settings as default box styles. Every new box created has the settings of the default box, but each box can be independently adjusted. These settings are performed easily, using icons in a requester.

PageMaker does not have the automatic borders, shadows, transparency, background, line spacing or justification between letters and words that PageSetter possesses. PageMaker does have a more flexible way of setting defaults, though. With PageMaker, you can design 'master pages'——pages designed just like other pages to be used as a template for new pages. This important concept is missing from PageSetter, which only has default boxes, page widths, margins and columns. This 'master page' ability brings consistency to your document and simplifies the use of graphics on every page. These master pages can also have automatically numbered pages. PageSetter has no way to automatically number the pages and the best way to work around the lack of 'master pages' is to use your previous document. This document is then loaded and the old text is replaced with new text.

Another design choice of PageSetter is its separation of tasks. With PageSetter, you are given a page layout program, a text editor and a graphics editor in one integrated package. PageMaker is a page layout program that allows typographical errors to be corrected and lines and circles to be drawn on the page.

PageSetter's text editor is fairly full-featured, but not as nice as a wordprocessor. The text editor can import TextCraft, Scribble! and plain ASCII text files. Search and replace, cut and paste, scroll bars, word wrap and a mouse-and-menu interface make up the editor. The editor cannot format the document. Formatiing, as well as displaying boldface, underline, and italics, takes place in layout. Symbols surrounding the plain text show which styles are used (For example, \bTHIS IS BOLD\b, boldfaces the text 'THIS IS BOLD').

I would like to see the text in the editor formatted as it is on the page and unformatted only when the text has not been placed on the page. Seeing the text with the styles in place and a faster and more responsive text editor would be a nice additions. It's important to note that PageMaker has no text editor, so the majority of your text editing must be performed in either MicroSoft Word or Apple MacWrite. Because the Macintosh does not multitask, so many switches are a nuisance when textual changes are needed.

continued...

The graphics editor in PageSetter cannot compare with other graphics editors available for the Amiga. . . but graphics editors on the Amiga are, without a doubt, the best available for any computer. The graphics editor does import IFF graphics files, allowing the use of any graphics editor available on the Amiga to produce graphics for PageSetter.

The graphics editor can be used for simple manipulations on existing images. These manipulations are performed using smooth draw, dot draw, line, rectangle, ellipse, spray can, fill, magnify, text, image brushes and 8 styles of pens. These are the only tools you'll need from this integrated graphics editor. The real editing can be done in a much more complex graphics program. However, PageSetter flaunts a much larger collection of graphics abilities than PageMaker's line, rectangle, rounded rectangle, circle and cropping tools.

Some general tools for page layout on PageSetter include: a grid snap feature, ruled edges, grid lines, three level magnification, quickedit button, article tracing ability, linked boxes and forward/back gadgets for boxes. The grid snap feature allows quick alignment and sizing of boxes on the page to produce an evenly- spaced grid. The quickedit button simply activates the correct editor for the currently selected box.

All these features are implemented in PageMaker, with the exception of article tracing and forward/back gadgets. The article tracing feature, used to follow the thread of a story as it jumps from column to column and through multiple pages, is sorely missed in Page-Maker. The quickedit button is used to jump into the correct editor. Simply click on a text or graphic and stab the quickedit button. PageSetter automatically invokes either the graphic or text editor with the selected item. Changes in the editor are reflected on the page when you exit.

Automatic reformatting, the most powerful ability of a page layout program, is common to both programs. Reformatting redistributes long text articles across many pages automatically, as boxes or columns are moved and sized. Without this capability, it would be nearly impossible to have multiple stories on each page.

Both programs also have a pasteboard feature. This pasteboard is a temporary area for scraps of articles, graphics and other odds and ends. A method for linking columns or boxes together is needed for stories to jump from page to page. This linking is provided in both programs, but it is very difficult to follow a story with PageMaker. PageMaker requires that you read the text on the pages and skip back and forth manually, just as you do when you read a real magazine. On the other hand, PageSetter has a next and previous box gadget that automatically skips through the story and highlights the region currently on the page.

Implementation of graphics is the major difference between PageSetter and PageMaker. PageSetter's graphics are strictly bit-mapped, while PageMaker's graphics are not. This difference means PageMaker's graphics are the resolution of the printing device. For example, a line on a laser printer uses 300 dots per inch (DPI) and produces a very smooth line, while a line on a dot matrix printer produces a jaggy, stair-stepped line. PageSetter's graphics are always the resolution of a 72 DPI dot matrix printer.

Actually, no matter which printer you use, PageSetter's text and graphics look the same as when you use a 72 DPI printer. This sameness is a major limitation of PageSetter. Gold Disk is aware of this limitation and has released a product called LaserScript. This product is used to print out PageSetter files on a laser printer, using the resolution of the printer and industry standard fonts such as Helvetica and Times Roman. LaserScript prints the box presentation and text in the maximum resolution of the printer using PostScript.

If you do not own a laser printer, PageSetter performs as well as PageMaker using dot matrix printers. While PageMaker can only use the Apple Imagewriter printer or PostScript printers, PageSetter can use any printer compatible with Preferences or any PostScript printer used in conjunction with LaserScript.

There is a caveat, though. PageSetter without LaserScript does not use the resolution of your Preferences-compatible printer. PageSetter manipulates a 72-dot per inch image, so that it fits within the resolution of your Preferences printer. I would like to see the program use whatever resolution your printer has, although this might be a difficult problem for program designers.

Fortunately, PageSettter is easy to use. The thin manual will be of little use after some experimentation with the program. The manual provides many good ideas for publishing, but does not answer most questions about the program. It does not have an index.

Overall, PageSetter is a very strong program with many capabilities not present in the Macintosh industry leader, PageMaker. With small improvements, Pagesetter could become better than PageMaker. . . in some ways, it already is better. In my opinion, PageMaker would have a hard time becoming as flexible and easy to use as PageSetter because major flaws in its design make it harder to use than PageSetter.

Another consideration is price. For the price, PageSetter beats PageMaker hands down. PageMaker retails for $495 while PageSetter retails for $150 (and another $45 for LaserScript).

If you do not own a laser printer and do not have access to one, PageSetter can provide you with high quality documents. PageSetter can even be used to do many things other than newsletters. Some uses that come to mind are posters, calendars, labels and just about anything else that is page-oriented.

Gold Disk is also selling a fonts disk that contains many fonts beyond those on the WorkBench disk. If you are planning on using PageSetter, this wider font selection would greatly improve the quality of your newsletters. With some small improvements in PageSetter's wordprocessor, I could recommend it as a wordproces-sor that allows you to use fonts. The print quality is much better than the quality printed by Notepad on the WorkBench disk. You must realize, however, that in its current form, PageSetter is merely a text editor.

Overall, PageSetter performs very well for the functions it was intended for. . . and can be used for many things it was not intended for. PageSetter is a good program and is very much worth the price—— it's a steal compared to its competition on other computers.

•AC•

PageSetter $149.95
LaserScript $44.95

Gold Disk
2179 Dunwin Drive #6
Mississauga, Ontario
Canada L5L 1X2
(416) 828-0913

Orders only (800) 387-8192

GIZMOZ Productivity Set 2.0

A collection of tools designed to make your Amiga easier to use

by Bob Eller
People Link Amiga*Bob

In the early months of 1986, Amiga owners were starving for software. The Amiga had been in the hands of users less than six months when Gizmoz Productivity Set was first released in February. Gizmoz was one of the first commercially available programs that let Amiga users productively use their machines. By the end of 1986, however, version 1.2 of the Amiga operating was released and many Gizmoz users found their old favorite showing signs of age.

Digital Creations, creators of Gizmoz, were ready for the program's first birthday with an upgraded Gizmoz Productivity Set version 2.0. The upgrade corrects AmigaDos 1.2 related problems, strengthens the features of each program and adds a few new goodies to make Amiga owners even more productive.

What are Gizmoz?

Gizmoz, for those unfamiliar with the package, is a collection of tools designed to make your Amiga easier to use. Those of you familiar with the MS-DOS world have probably encountered a similar program, SideKick. SideKick is memory-resident utility which provides a notepad, calculator, appointment calendar, and phone dialer while running other PC programs. ("Memory-resident" is a term used to describe MS-DOS programs that pretend to multitask.)

Gizmoz, like SideKick, provides Amiga owners these basic utilities and much more. Since the Amiga can run more than one program, Gizmoz utilities are available to use with any program that properly multitasks and shares the Amiga's resources.

Digital Creations has grouped their utilities in five categories. Organizers to help you keep track of important information, accessories, audio visual programs, Workbench tools, and calculators.

Organizers

Gizmoz organizers are the some of the most useful programs in the package. The organizers include:

MEMOPAD

Memopad is a text editor, comparable to the Notepad included with the Amiga WorkBench, which can be used to write and edit memos, letters, small programs, and other simple documents. Unlike Notepad, Memopad includes many keyboard shortcuts to use in editing your work. You can run several memopads and cut, copy and paste between each memo.

Memopad has some advantages over the standard Notepad. Memopad is a much smaller file and, as a result, takes up less space on your Workbench disk and loads more quickly than Notepad. If you like to use different font styles in your documents, you'd better stick with Notepad. Memopad supports only the system default font.

ROLLODEX

If you've ever needed to find that slip of paper with an important phone number, then you'll appreciate the value of the rollodex program. Rollodex allows you to design an index card to keep track of important data. Each card can contain five rows of thirty-six characters. The cards you create can also be searched and sorted to suit your needs. If your card contains a phone number, Rollodex will dial it for you using your modem. Included with Gizmoz is a sample name and address file to get you started.

Rollodex can also serve as a small database. There are no fixed fields or formats for the cards. This freedom allows you to enter whatever information you wish to store in any order. Searches and sorts are executed by typing the text "string" you want to find into a workbench requestor. Instead of names and addresses, you could create a mini disk catalog or a database of any other information you would like to store and retrieve.

CALENDAR

Calendar provides a 200 year perpetual system for the years 1900 to 2099. You may keep up to fifty lines of appointments each day, with a reminder of your appointment provided by the Amiga if you request it. Each monthly calendar highlights a day with an appointment by showing the date in white.

Calendar also supports the same free-style format found in the Rollodex program. Appointments may be entered in any order, alleviating the need to rearrange your data if you enter an additional engagement.

BLACK BOOK

The Black Book tool is used to print and format your Memopad, Rollodex and Calendar data. In addition to helping manage your printouts, Black Book creates mailing labels and Rollodex cards on your printer when used with your rollodex files and the suitable continuous forms.

HOTKEY

Do you find yourself executing the same keystrokes over and over again? Hotkey allows you to define several keystrokes as one keystroke or "macro". You can create several different hotkey files for each of your favorite programs and load them as needed.

Accessories

The Accessories drawer holds a grab-bag of Gizmoz, including:

TERMINAL

Terminal provides a full-featured tool for accessing various telecommunication services with your modem. The terminal package includes several of the most popular terminal emulations (VT-100, ANSI, Televideo, TTY, ADM-3A, and Lear-Siegler) and file transfer using Xmodem, text, and simple modem. The terminal package also includes an Amiga Binary protocol which transfers any icons associated with the selected file. Terminal also allows you to save incoming text for later viewing.

There are many good Amiga terminal programs in the public domain. Gizmoz terminal lacks some of the features, such as a phone book for storing often used numbers, that can be found in these public domain programs. Terminal does contain some emulations that would be of interest to Amiga users who access these specialized computer systems.

In addition, Terminal's ability to transfer icon information when using the Amiga binary file transfer is a feature that, to my knowledge,

continued...

is not supported by any other Amiga terminal program. Other terminal packages require that you either send two files or use a file utility to package both files as one, or the terminal program creates a generic icon for the file you receive. Since many customized icons are available, transferring the icon with the program is a plus. This option is only available when communicating with another Amiga using the Gizmoz Terminal. None of the commercial telecommunication services or Bulletin Boards support the Amiga Binary protocol.

COMPRESS
One way of shortening the amount of time spent in sending files over the modem is to compress the file before sending, and decompress the file to its original form at the receiving end. Compress uses a version of the Huffman compression algorithm to perform the task.

Most Amiga file compression is done with the public domain archive program Arc. Arc is widely supported in the Amiga telecommunications community and is file compatible with the IBM PC version. To test the Gizmoz Compress versus Arc, I chose a 19.7K text file. Arc compressed the file to 10.2K, while Compress yielded a 12.3K file. The only benefit to using Compress, therefore, is its use of the Workbench interface. Arc requires that you use the CLI.

ENCRYPT
If you need to keep prying eyes from your data files, then Encrypt is your program. The encryption is based on a key string entered by the user. If you forget the string, you won't be able to unencrypt the file. Hmmm, sounds like something the folks in Washington could use! A typical Amiga user won't find himself using this program too often.

FREELIST
Freelist gives a graphic representation of the Amiga's memory use. Memory can be display in either of two ways, free or changed. The changed memory option allows you to snapshot your memory usage, perform a task, and view the change to the memory configuration since the snapshot.

There are several "Freelist" type programs in the public domain and in the Tools drawer of the Workbench 1.2 Amiga Extras disk. The "Freemap" program on the Extras disk will not work with memory expansion, while the Gizmoz Freelist will look at all the memory you add to your Amiga. If you are using a memory mapping type program on a regular basis, then the Gizmoz Freelist will be useful to you.

One additional note, if you use the ASDG protected ram disk, VD0:, Freelist does not appear to work properly! I grabbed the slider switch while running Freelist and scrolled through the memory map. As the slide reached the bottom of the memory display, the machine crashed. After I removed the protected ram disk, Freelist ran without a hitch.

SETPRIORITY
This program is not for the timid! Setpriority allows you to directly affect the priority of tasks running on the Amiga. You could, for example, increase the priority of a task to speed its execution. Be warned, unless you are familiar with how task priorities are set, you run the risk of losing data and visiting the Guru! Most Amiga users will have limited use for this program.

POPUP
Popup is one of the handiest tools available in Gizmoz. Popups provide knowledge at the click of a mouse button. The Popup screen can be pulled up when needed, or pushed to the bottom of the screen for later use. Included are popups for Microsoft Basic, AmigaDOS, and ASCII conversion. You can also create your own Popup containing the information you need to have handy. The program includes a search option to quickly locate needed data.

AudioVisual
Using a computer shouldn't be all work. The AudioVisual drawer supplies several practical tools and some fun ones, too.

GRAPH
Graph can quickly help you put the finishing touches on your latest project. Graph uses text scripts, like those created by the memopad, to design bar, column or pie graphs. Graph will customize the color of the graph segments and the placement of the labels. You can save the graph as an IFF file and print or modify it further using any IFF compatible graphics program.

CUCKOO CLOCK
A variation of the standard Workbench clock, has a cuckoo to sound off the hours. The clock includes an alarm function to keep you on time.

ANNOUNCE
Announce allows you to play with the speech functions of the Amiga. Type in a phrase and the Amiga speaks it. You can modify the voice to suit your fancy. A valuable feature to programmers is the ability to translate text to phonetic phrases.

Gizmoz Announce is a rewrite of the public domain program SpeechToy. The only addition is an announcer whose face appears in a window announcing your text. SpeechToy was a very early Amiga demo program and, unlike Gizmoz Announce, may have problems when used with version 1.2 of the operating system.

SUPERLIFE
This accessory is an implementation of one of the first computer games, LIFE. LIFE, as the name implies, depicts the life of a cell and its colony over the generations. SuperLIFE is a very fast version and can create up to 60+ generations a second.

Calculators
The calculator's drawer contains three models: scientific, financial and programmer. The scientific model contains a complete set of scientific functions such as sin, arcsin and log. The financial calculator can calculate interest, payments, future and present value and more. The programmer's calculator contains both logical and basic mathematical functions. Each of the calculators can save and print a tape of the entered keystrokes and results for later use.

The Gizmoz calculators are the best I've seen. Each contains a full implementation of the features I'd need to program, do scientific analysis or get my finances in order. Digital Creation's programmers spared no detail. When you click a calculator's key, the key depresses on the screen.

Benchtools
The Benchtools drawer is new on Gizmoz 2.0 and contains:

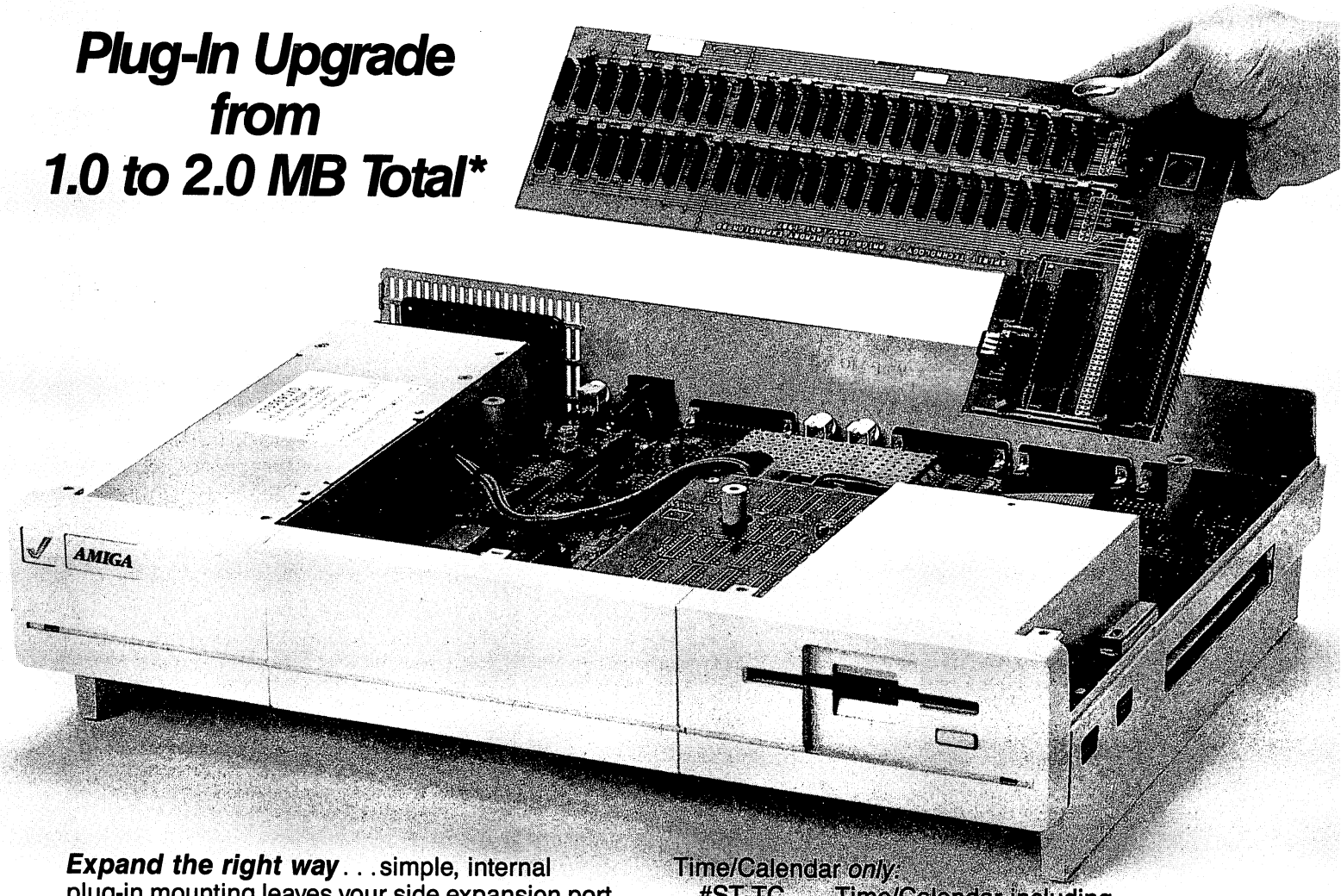
FASTPREFS
Fastprefs is a utility for quickly altering the system preferences. With Fastprefs, you can create and save data files containing the changes you wish to make. When you want these preferences implemented, just double click the data file icon and the changes are installed. Included is a sample of the favorite WorkBench settings at Digital Creations.

One user of Gizmoz 2.0 reports that he was upset when he found that the only way to get the original preferences back, after trying the Digital Creations settings, was to reboot the machine. The moral here is to create a Fastprefs file with your favorite preferences first. The sample may not be to your liking!

continued...

AMIGA INTERNAL MEMORY EXPANSION Plus TIME/CALENDAR

Plug-In Upgrade from 1.0 to 2.0 MB Total*



Expand the right way. . . simple, internal plug-in mounting leaves your side expansion port free to add other peripherals. Also, the internal Time/Calendar does not use a joystick port.

Memory Expansion Features:

* Zero Wait-State
* No Cuts or Soldering Required
* Full Auto-Configuration
* Lithium Battery Back-Up for Time/Calendar

ORDERING INFORMATION:

DRAM Memory *with* Time/Calendar:

#ST-05	0.5 MB	$349.50 List
#ST-10	1.0 MB	$499.50 List
#ST-15	1.5 MB	$599.50 List

*Memory expansion from 1.0 to 2.0 MB includes AMIGA 1000 512K RAM.

Time/Calendar *only*:

#ST-TC Time/Calendar including Battery Back-Up $59.50 List

ASK ABOUT increased speed with the *new* 68010 Processor

VISA and Mastercard Welcome

CALL TOLL FREE: 1-800-433-7572

Factory direct: 1-801-485-4233

DEALER INQUIRIES INVITED

SPIRIT™ TECHNOLOGY

220 West 2950 South • Salt Lake City, Utah 84115

JOYMOUSE
Sometimes the mouse isn't the best tool for the application. Joymouse allows you to use a joystick plugged into the second mouse port, instead of your Amiga mouse. Joymouse doesn't disable the normal mouse, so you are free to switch as needed. Should your regular mouse decide to fail, Joymouse could also provide a temporary replacement.

QUICKMOUSE
QuickMouse accelerates the movement of the cursor when you move your mouse quickly. This means you don't need to move the mouse as far to get an equal amount of cursor movement-- a real boon to those with cramped desks. If you move the mouse slowly, however, you retain the accuracy you would normally expect. I liked QuickMouse so much that I've added it to all my workbench disks and startup-sequences so that it's automatically installed.

ICONMAKER
Graphics help make the Amiga such a great machine. You can use Iconmaker to convert your favorite IFF picture to an icon usable on the Workbench. You could, for instance, make an icon for a picture by using the picture as the icon. This is an easy way to track picture files.

Common features
Gizmoz 2.0 includes two standard enhancements for all its tools. They have added a standardized file requester for programs, such as Memopad, that access files. The requester displays a list of the available files. Files are retrieved by pointing at the file and double clicking the left mouse button.

If you are tired of Workbench clutter, then Gizmoz 2.0 can help. All the tools include a "small window" function. Double clicking the right mouse button will shrink the window to a small strip. Double clicking on the strip will expand the window to its original size.

Upgrades
Gizmoz is available at a retail price of $69.95, although a quick check of Amiga software dealers showed the program selling at far less than retail. Before buying Gizmoz, determine which version of the program the dealer is offering!

If you own the original Gizmoz package, an upgrade is available for $15 plus shipping and handling. Contact Digital Creations at (916) 344- 4825 for complete details.

Included with Gizmoz is a manual with complete instructions on each program, as well as some general tips on using your Amiga more efficiently. Best of all, Gizmoz is unprotected. You can create specialized Workbench disks and use only the Gizmoz you need to have handy for a specific application.

If you are a new Amiga user, Gizmoz can get you off on the right foot. Even if your an old hand, there is sure to be something of value for you in Gizmoz.

I've found that the Gizmoz I use most are the HotKey, Rollodex, Calendar and accompanying Black Book from the Organizers drawer; Popup from the Accessories drawer; Graph from the Audio Visual drawer; all of the calculators; and Fastprefs, QuickMouse and Iconmaker from the Benchtools. That's twelve of the twenty-two programs included in the package. At a cost of $69.95, that translates to $5.83 per application. SideKick, which offers only a notepad, calendar, simple calculator, phone dialer and ASCII conversion table, retails for $84.95 or $16.99 an application. In comparison, Gizmoz is a good deal! Decide which of the features you could use and make this comparison for yourself.

•AC•

KickWork

combination Kickstart and Workbench

by Harv Laser
People Link CBM*HARV

The Amiga lends itself to more customization than any other microcomputer. Turn on any Commodore 64 and you'll see the same thing: a blue screen with white text and a flashing cursor.

The Amiga is quite different when turned on. I have seen many dozens of Amigas and watched owners of varying levels of proficiency power up their machines. It seems like no two of them are alike. Beginners usually stick to the Workbench environment, especially if they've never owned another computer. More informed users change the Preferences pointer, modify their startup-sequence file to open a CLI window or start a program automatically.

No matter how individualistic they were, all Amiga 1000 owners had one thing in common: when they turned on the machine, they had to insert a Kickstart disk. All Amiga 1000s have circuitry known as the "writeable control store," or WCS. This is 256K of RAM used to hold the Kickstart program code. You have no access to this memory; it is not under your control.

Under normal conditions, your computer can only read from this special WCS memory, but not write to it. This protects the Kickstart program from being corrupted. You don't want your operating system to be corrupted, many bad things can happen if it is.

The new Amiga 500 and 2000 computers differ. They don't need a Kickstart disk when turned on. The code which the Amiga 1000 loads from the Kickstart disk is already permanently "burned into" ROM chips, and when powered up, the "Insert Workbench" screen appears: Kickstart is not asked for! This has advantages in that power up is quicker and one need not hunt for that pesky Kickstart disk which sometimes has a mind of its own as it finds its way into places on your desk you didn't even know existed.

The disadvantage with Kickstart in ROM is that you are "locked into" the operating system supplied with your machine. The A500 and A2000 have a slightly modified Amiga operating system version 1.2 in ROM. Suppose Commodore issues newer versions with more and and better features? What to do? Owners of these newer machines may be faced with having to open their computers and replace the chips which contain the Kickstart code - the Amiga 1000 owner need simply buy a new Kickstart disk which is what happened when Commodore upgraded from version 1.0 to version 1.1, and then again to version 1.2.

So, we can look at having Kickstart on disk instead of on chips as an advantage or a disadvantage.

One company sells a hardware kit that lets an Amiga 1000 owner have the same capability as the Amiga 500 or Amiga 2000: by installing a custom circuit board into his Amiga, the Amiga 1000 owner can also avoid the "insert Kickstart" prompt and the necessity to boot with two disks. He even gets to use the special RAM chips that housed Kickstart for an extra 256K of RAM memory for running programs.

However, the Amiga 1000 has no "slots" and must be opened up to make this installation. The average Amiga owner might not feel like disassembling his computer and doing precision soldering work inside, if he is even capable of it, nor paying a technician more money to do the job for him. This "Kickstart in ROM" hardware modification could easily cost a couple hundred dollars to purchase and have installed.

There is another solution to the Kickstart-on-disk situation. There is a way to put both Kickstart and Workbench on one disk. Inserting this disk into the internal drive and powering up your Amiga 1000 will load the Kickstart code into the WCS and then boot up Workbench just as though you had done the regular two-disk routine.

Actually there are two methods of reaching this same goal. One of them is virtually free, and involves locating two pieces of software: A copyrighted but freely distributable program called "Makeboth", written by Alonzo Gariepy of Toronto, Ontario, Canada. It is available on Fish disk 36 and on most BBSes and commercial networks.

Makeboth includes programs and documentation which will let you modify a copy of your Kickstart disk so that it contains both the Kickstart and Workbench portions of code. When inserted into your internal drive and power is applied to your Amiga, Kickstart will load followed immediately by Workbench. There is no need to change disks.

There are two small problems in using Makeboth, however. Makeboth requires you to have a copy of a program called "DiskEd" which is supplied on a disk which Commodore sold to registered Amiga software developers. Legally, DiskEd is licensed only to those developers. It is not public domain nor distributable software, so you won't find it on public domain disk collections nor on the commercial services, and if you're not a developer, you're not even supposed to have it.

Another hassle is that the disk needs to be "armed" after each use. When the Kickbench/Makeboth disk is first used, the Amiga thinks it's a Kickstart disk. When it has finished its job and left you at your CLI or Workbench screen, the Amiga thinks it is a Workbench disk. In order to use it as a Kickstart disk the next power up, you have to remember to "arm" the disk by running another program to reset the disk. Some users report that the re-arming command is not always foolproof. This can lead to frustration. There is no free lunch. The convenience of a one-disk-boot computer is offset by re-arming your disk before turning off the power.

A company called Amigo Business Computers has come up with another product to perform a similar function as Makeboth accomplishes, but with a more elegant end effect, and it is much simpler to use. You need not go "hacking" into your Kickstart disk, and there is no need to try to get your hands on a program which only developers are supposed to have Kickwork is the product and its name pretty much explains what it is.

continued...

Kickwork is a combination Kickstart and Workbench disk. You buy it, and it's ready to use. Simple as that. All the drudgery and inconvenience of having to cobble up your own "dual disk" has been done for you. For a retail price of $29.95, you can save yourself much trouble and reach the same goal.

Kickwork, when be inserted in your Amiga 1000 powered up, will load the Kickstart code and then immediately load Workbench from the same disk. In its default configuration, straight out of its package, you are left looking at what appears to be a pretty standard Workbench screen with a new color scheme a little "hand" pointer, and a custom disk icon. Your Workbench's menu bar will also announce the fact that you are using Kickwork.

Naturally, Kickwork comes to you unprotected: you can make as many copies as you need for your own personal use. In fact, the manual explains that this is a good idea. By deleting some files from Kickwork and moving a word processor or spreadsheet program onto it, and modifying the disk's startup-sequence file, you could create a complete "turnkey" system that a novice could use. Kickwork may be booted with the write protect on its disk on.

Kickwork would also be an ideal product if you live in a part of the country prone to voltage fluctuations and electrical storms. If Kickwork was in your internal drive, and you suddenly lost power for a moment, the computer would reboot automatically. Operators of Amiga-based BBSes would greatly benefit from Kickwork. If a power drop or failure hits while the BBS software is running, when power returns, the BBS would reboot off the Kickwork disk.

Kickwork does have some limitations, but they are by no means worrisome ones. Including the Kickstart code on the disk eats up about 256K of space on an 880K disk. This leaves a little more than 620K of space for the Workbench, AmigaDOS commands, printer drivers, fonts, etc. In order to get everything on one disk, the Kickwork disk does not contain all the files found on a standard Workbench disk. You can strip other unneeded files from the Kickwork disk to suit your situation.

The program has written a modified version of the AmigaDOS 'info' command to replace the standard 'info' on Kickwork. The Commodore supplied 'info' command will report Kickwork as being a Kickstart Disk whereas the new 'info' command will indicate a more "normal" reading of the disk's name, blocks used, etc.

The AmigaDOS 'diskcopy' has also been replaced with a special version to accomodate the altered disk structure of Kickwork. It is important to note that your copies of Kickwork must contain these modified 'info' and 'diskcopy' commands, as the standard Commodore-supplied versions of those two commands will not function correctly with Kickwork.

By copying your Kickwork disk with the special Diskcopy command, moving files, changing the startup-sequence, and customizing the disk to your needs, you should be able to create boot disks with any popular application that is not copy protected. Copy protected software will pose a problem and such disks are better off booted directly from their manufacturer-supplied media. If the manufacturer offers an unprotected version of their software, then you can most likely combine it with Kickwork.

There is one other problem when using Kickwork: if one day you decide to boot your system using a regular Kickstart/Workbench two-disk method, you won't be able to use your Kickwork disk at all. AmigaDOS won't recognize it as a DOS disk. You must reboot from Kickwork in order to use it. I consider this to be a very minor inconvenience. If you're going to go to the trouble to purchase Kickwork you'll probably be booting from it all the time, in which case all of the files on its disk will be available to you.

Kickwork is supplied with a short but complete and understandable manual. The more adept you are at using CLI, the more you'll be able to accomplish with Kickwork. Since the Amiga is so completely customizeable to each owner's tastes, Kickwork is a painless way to shorten the powerup sequence without resorting to hardware modifications.

I asked William Teller, president of Amigo Business Computers, how he planned to handle upgrades to Kickwork, should Commodore release more upgrades to the Amiga operating system. Teller said his company's policy is to upgrade original purchasers of Kickwork for a "nominal" cost to cover mailing and handling charges, and that Kickwork will be upgraded if Commodore upgrades. More information can be obtained directly from Amigo.

Due to legalities, and since Kickstart is copyrighted by Commodore, you must already own the Commodore Amiga Enhancer package to purchase Kickwork. If you haven't yet purchased your 1.2 package you may buy it directly from Amigo with Kickwork included. Recently, Amigo has added another method of purchasing their program: they will sell you what amounts to a "Kickwork construction set" disk which can be used once to create a copy of Kickwork from a standard set of Kickstart and Workbench. The special code which performs this task will then erase itself, leaving you with an original Kickwork disk which you can copy.

•AC•

Avatex 1200

THIS MODEM WORKS WITH ANY AMIGA.

- Direct Connect
- External AC Adapter
- 8 LED Status Indicators
- Auto Answer and Dial
- Hayes Compatible AT Command Set
- Bell 103/212A Compatible
- Tone or Pulse Dialing
- Telephone Connection Jack
- Standard DB25RS 232-C Connector
- FCC Approved

COMPUTER SHOPPER Best Buy OF THE YEAR

MODEM

$85

FREE! Communication Software & Compuserve Access Time with each MODEM.

We give you a great price . . .
+ TOLL FREE ASSISTANCE
+ IMMEDIATE DELIVERY
+ NO CREDIT CARD SURCHARGE
+ NO SERVICE OR HANDLING FEES
+

We lowered our price.

Thousands of people are now enjoying this high quality Modem. It is full of features and works with any computer system. We are offering this special price for a limited time only.

FREE SHIPPING!

MEGATRONICS

To Order
CALL FOR FREE CATALOGUE
CREDIT CARDS VERIFIED FOR YOUR PROTECTION

800-232-6342
INSIDE UTAH (801) 752-2642 FAX (801) 752-6752

We'll beat any advertised price.

MEGATRONICS, INC., P.O. BOX 3660, LOGAN, UTAH 84321

Diga! Telecommunications Package

reviewed by Steve Hull

GEnie: LightRaider
People Link and BIX: St.Ephen

There are many adjectives I can think of to describe a telecommunication program. Until Diga! came along, "controversial" would have been the last to occur to me - or, I suspect, to Aegis Development. Yet, it is hard to imagine that such a simple program could be the subject of so much controversy in the Amiga community.

Long promoted in glossy handouts, Diga! (Spanish for "speak to me") sounded great. Aegis promised an extravagant list of standard features, such as executable scripts, remote capability and multiple terminal emulations. Aegis then added the coup de grace: DoubleTalk, a new proprietary transfer protocol that allows two people with Diga!-equipped computers to send files in both directions simultaneously, while carrying on a real-time chat.

Some Amiga owners could hardly wait for Diga! to be released. In fact, they didn't wait for it to be released. Shortly after Aegis issued an extremely limited number of beta test copies, bootleg Diga! disks popped up across the country. Aegis clamped on a tight security lid and set to finish the job of ironing the bugs out of Diga! for its projected second quarter 1987 release.

In early summer, the program was released. Ah, but all was not bright. Within my first hour of using Diga!, the program locked up three times. No guru, but no way to "quit" the program and free up memory. Deb Christensen, sysop of GEnie's StarShip Amiga, dourly noted that all she had to do to get the program to crash - full guru - was attempt to pull down a menu. On Usenet, the author of a competing telecom program posted a detailed Diga! bug list, crowing about having such "easy competition." As if to add insult to injury, users of the bootleg pre-release Diga! began sending up a hue and cry over bugs in their version. Suddenly, this review looked like a can of worms.

The unfortunate by-product to all the controversy is it has far-and-away overshadowed Diga!'s many strong points, and that's a shame. In producing Diga!, Aegis has not marketed telecommunications' Holy Grail. In fact, in some areas, the designers made, what I consider to be, major miscalculations. The fact remains that Diga! has got enough power and features - and, uniquely, potential to earn its place in any telecommunicators' arsenal.

Indentity crisis

Though Diga! has some software bugs, its greater problems result from poor documentation and an imprecise marketing approach that attempted to present the program as being all things to all people. In short, Diga! is a product with an identity crisis.

On one hand, the documentation reassures newcomers to telecomputing that Diga! is "very easy to understand and use." The manual begins with a short introduction to the basics of telecomputing - what is meant by terms like duplex, parity and baud rate, for instance. Diga! also aids the beginner through full use of the Amiga's user-friendly features, and employs amenities like the "fast" menu to make everything available quickly.

The paragraph that begins with a pitch to the new user, ends with a pitch to the telecommunications pro, promising "the power and the features" such users require. To deliver, Diga! includes such advanced features as a Tektronix 4014 graphics terminal emulation, 132-column by 50-line overscanned display, powerful script functions, and a "remote" function that turns your Amiga into a small-scale BBS while you're away.

So far so good. However, in its effort to satisfy the power user, Diga! short-changes the recreational telecom user, by omitting such basic features as auto-redial and a split-screen display for use in real-time conferences. In addition, the documentation is incomplete, sometimes incorrect and includes the kind of gotchas that perplex experienced telecom users, to say nothing of beginners.

Should the experienced user log onto the VAX mainframe at work, using Diga!'s VT-100 emulation, he will find the program does not appear to support the PF programmable function keys used in the system text editor an essential function. In reality, Diga! does support the PF keys but the manual holds no clues.

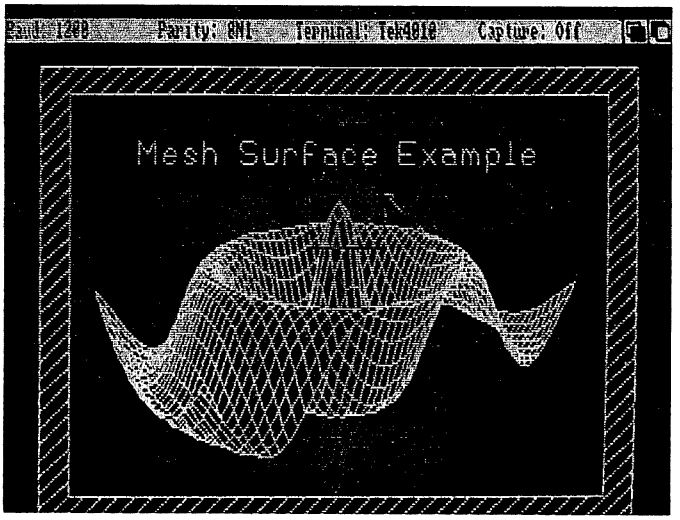
Program size is another factor that comes into play when software tries to be "all things to all people." Diga! takes up 163K on the disk. When installed, the program consumes 274K before you get the phone off the hook.

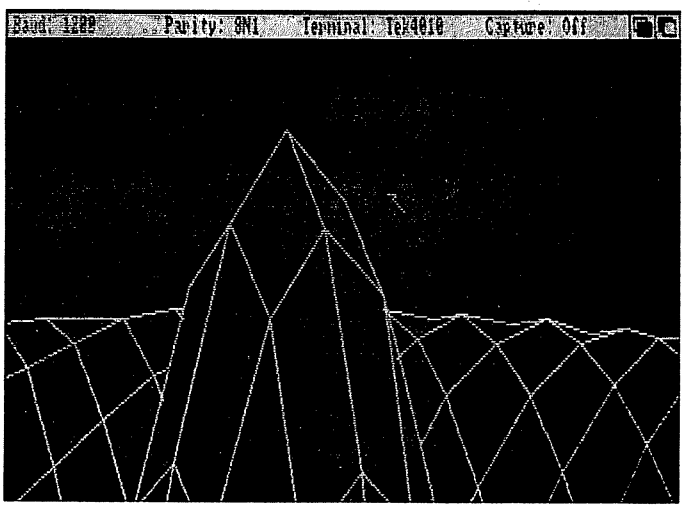
On a standard 512K Amiga with one external drive, that leaves about 180K for multitasking - not a lot of room on a graphics and sound-oriented computer. Aegis claims that almost all multitasking problems encountered with Diga! can be solved by adding additional memory.

Documentation troubles

Diga!'s greatest liability, however, is not the software itself. The documentation is a travesty. On a casual glance, it looks decent enough. It has clear subgroupings large, clear type and many illustrations. The more you get into the program, though, the more the shortcomings become apparent.

For example, take the problem of the "missing" PF keys in the VT-100 emulation. You log onto the host system successfully and prepare to edit some source code using EMACS. After fumbling with the Amiga function keys and getting nowhere, you turn to the index of the Diga! user's manual. No index.





Diga!'s Tektronix 4010/4014 emulation allows you to zoom on any portion of a drawing on-line. Also included: a stand-alone utility to convert captured Tek graphics to Aegis' Draw Plus CAD format.

Next, you check the table of contents for anything that looks like VT-100. Nothing there, not even a subheading for "emulations." Out of desperation, you begin paging through the manual. On page 50, your eyes catch the paragraph heading, "Terminal Emulations." Bingo! Eagerly, you flip pages for some information on VT-100 functions. Eagerly... and in vain. Here is the Diga! manual's sum total documentation on both its VT-100 and VT-52 modes: "The VT series are standard DEC emulations..." That's it, bucko. Happy hacking.

The VT-100 PF keys are accessed through pressing the CTRL key, in conjunction with the 9, 6, 3, or decimal from the numeric keypad, plus SHIFT-comma. This feature and others are absent from the manual, and were released by Aegis as part of a public-domain "Diga! Tricks" file. See the sidebar to this article.

Macro key troubles

The manual also sets users up for a fall in its description of the procedures for using control (CTRL) characters in macro keys. To enter a control character in a macro key string, you must hold down the CTRL key while pressing another key. For example, ending a macro key definition with a CTRL-C sends a "break."

Unless you follow the manual, that is. On page 56 of the manual, Diga! users are told to enter the two characters, "^" and "C", to send a CTRL-C from a macro key. It doesn't work. Entering a "^C" in a macro key results in a caret-C being printed when you press that key. To enter CTRL-C, you must press CTRL and C, at the same time. All control characters display as a square box, which makes it impossible to tell a CTRL-G (bell) from a CTRL-M (carriage return) by examining the macro key definitions.

You may have gotten the impression by what I have said that I don't like Diga! Not true. I find myself appreciating Diga!'s many options more and more each time I use it. In fact, it is the program's very richness and flexibility that caused it some early bad press. People were booting the disk without even a glance at the docs, trying to make Diga! work like their old terminal program and getting into trouble.

Phone database

A classic example of this approach is the way Diga! keeps track of phone numbers. Like other Amiga telecom programs, Diga! allows users to define separate terminal configurations for each phone number, automatically reconfiguring your terminal when you tell it to dial the number. Unlike other programs, Diga! keeps scripts, configurations and phone numbers straight, by putting them in separate directories.

Selecting PHONE BOOKS from the PHONE menu pops up a requester with one Phone Book entry displayed. One entry may hold quite a bit of information; besides the name and phone number, there are three more blank lines for an address or comments. In addition, there are two lines to enter the name of the script for that phone number and the corresponding configuration.

The Script: and Config: entries require a full path name (i.e., CONFIGURATIONS/BBS.config, not simply BBS.config) to work correctly, which tripped up a lot of early users. Diga! will supply the path name automatically, if you click on the Script: or Config: words next to the text entry boxes—but these are not "obvious" requesters.

The Phone Book requester allows you to set and save baud rates for each number, as well as whether the number is voice or modem. The alphabet runs along the bottom of the requester, allowing you to quickly advance to the first terminal, beginning with a particular letter by clicking on that letter. Arrow gadgets at either end of a slide-bar allow you to step through Phone Book entries in alphabetical order. The Phone Book also allows you to print any or all entries.

Have it your way

Perhaps Diga!'s most powerful and underrated feature is the extent to which it may be customized to fit the requirements of any BBS or host system. By calling up the Fast Menu, you may set baud, parity, lines and columns per screen (including overscan), protocol, handshake, echo and end-of-line options with a series of rapid mouse-clicks... and this is only the beginning.

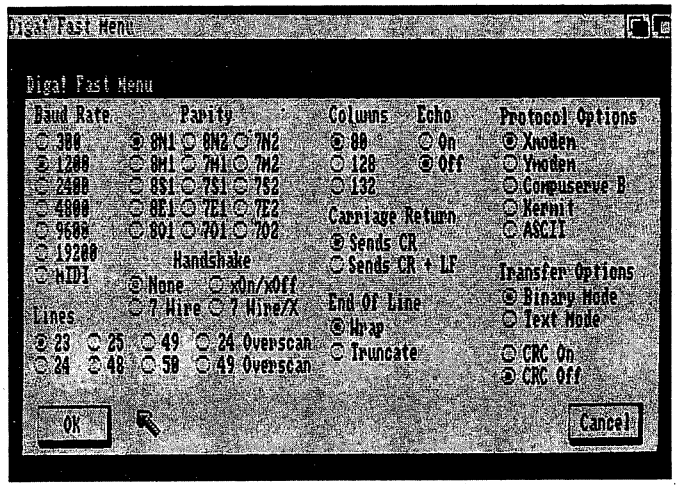
Diga! allows you to set 50 separate macro key strings, and each macro key may contain up to 80 characters. You can set the size of the capture buffer from 4K to 512K. You can specify a terminal emulation, bell options, screen colors and remote setup. You can even design custom screen configurations using any 8x8 or 5x8 font and add them to your pull-down menu options.

Everything is saved when you save the configuration. With Diga!, you can reconfigure from a 132-column, 50-line VT-100 terminal for work, to a 40-column, 24-line ANSI terminal (for those nostalgic visits back to your old C-64 BBS) with a single click this act simultaneously recalls all corresponding scripts, parameters and macro keys.

Diga! begins with complete implementation of the Intuition interface, then improves on it with nice touches, like hot keys for nearly every function. Diga! is fast, integrating

continued...

Charlie Heath's FastFonts routines into the screen code. The display easily keeps up with 2400-baud text. If you have expanded memory, you'll find Diga! multitasks better than most of its competition; it does not constantly poll the serial port, lessening its demands on the CPU. Anyone who has attempted to run Ed while using Micro Systems Software's Online! 1.0 can appreciate what I mean.



It wouldn't be an Aegis program without a Fast Menu. This one allows you to set up the major communications parameters with a series of quick mouse-clicks.

Flexible transfers

Diga! affords more file transfer capability than the average user will probably need - XModem (both standard and CRC), ASCII, Compuserve-B, server mode Kermit and YModem. The YModem is an advanced protocol that "downshifts" from 1024 byte blocks to 128 byte blocks, if it experiences an excessive number of transmission errors. Baud rates from 300 to MIDI speeds (approximately 31,000 baud) are supported.

One protocol, notable by its absence is WXmodem, a speedier variant of XModem supported by People Link and Delphi. The WXmodem transfer protocol is at the top of the list of new features for Diga 1.1 - more on the upgrade later.

Diga! takes even the standard configurations one step further. Both YModem and Xmodem allow batch transfers; a pretty neat trick whereby you call up a host system that supports batch transfer, click off as many files from the menu as you want to send and hit OK. Diga! takes over from there, transferring each selected file one at a time, automatically. You don't even have to type a filename. It is this kind of convenience that makes Diga! an attractive choice to people who spend a lot of time on-line.

DoubleTalk

Diga!'s biggest claim to fame is a completely new protocol designed by Keary Griffin and Rocky Stargel. The DoubleTalk protocol allows two Diga!-equipped terminals to simultaneously send and receive files, while the operators at both ends participate in a chat.

How does it work? Like a champ. In most one-way transfers, DoubleTalk is probably not the protocol of choice; it's a bit faster than XModem, but YModem smokes it. See Figure 1 for comparisons.

Tests were conducted at 1200 baud, using a 51,864 byte archived file. The local test transferred files across town. The long distance test stretched from south Texas to central California.

DoubleTalk(1) tested the speed at which a file could be sent one way with no chat. DoubleTalk(2) tested the protocol by sending the identical file both ways simultaneously. DoubleTalk(3) added chat, though it is difficult to precisely measure the impact, since DoubleTalk ignores the CAPTURE command. Had we been able to CAPTURE the chat, it would have been possible to measure the size of the total chat in bytes.

Figure 1
Relative Protocol Speed Comparisons

Protocol	Local	Errors	Long-Dist	Errors
XModem	7:55	None	8:11	None
YModem	7:10	None	7:13	None
CIS-B	7:38	None	7:55	None
DTalk(1)	7:44	None	7:46	None
DTalk(2)	7:51	None	7:57	None
DTalk(3)	8:16	None	8:11	Two

Given a clean phone line, there is virtually no speed difference between one-way and simultaneous two-way Doubletalk file transfers. Adding a chat slows things down a little, but not much.

Precisely how Diga! accomplishes this connection is covered in a separate sidebar. Suffice to say from a user's point of view, DoubleTalk is a gas! I have always hated the enforced silence imposed by standard protocols when uploading files to a friend. With DoubleTalk, conversation is hardly interrupted.

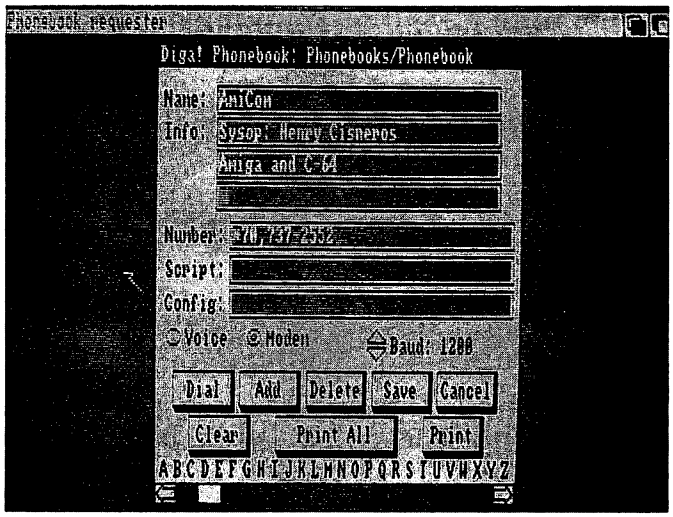
Widespread acceptance of the DoubleTalk protocol could turn telecommunication on its ear. Imagine calling up a local BBS and downloading a file while browsing the message base. According to Aegis, at least one company is integrating support of DoubleTalk into their Amiga BBS software. How fast DoubleTalk spreads will have a lot to do with Aegis' willingness to "go public" with the nuts-and-bolts of the protocol; at this writing, they're keeping it to themselves.

Software with slots

Aegis' method of integrating support of various terminal emulations marks an important crossroads in software design, and may, in fact, be shape of things to come. As opposed to conventional emulation support coded integrally into the program, Diga! loads its terminal emulations from separate files. Diga! is, as Aegis' VP for Technical Development, Bill Volk calls it, "software with expansion slots." Aegis has uploaded technical guidelines for the design of Diga! emulation files, as well as the Aztec C source code for the terminals included with Diga!, on several information services for public-domain distribution. Knowing the kind of technically-hardcore user the Amiga attracts, we shouldn't have to wait long before some interesting new emulations start showing up.

This new way of doing business has not been without glitches along the way. One of the most common sources of system lock-up results from switching from one terminal mode to another, selected from the Phone Book. Diga! defaults to TTY. This switch may be changed by the user. I have found that trying to dial an ANSI-configuration through the Phone Book results in a system lockup more often than not.

Besides the VT emulations, Diga! supports the ANSI color terminal emulation, Tektronics 4010/4014 and an undocumented "Talk" emulation, which uses the Amiga's standard speech synthesis hardware to read whatever rolls across the screen. As you may guess, "Talk" is an interesting novelty, but doesn't begin to keep up with 1200 baud. I decided to terminate my test of "Talk" the first time I called a board that used a row of sixty asterisks to border a menu. You haven't lived until you've heard your Amiga drone "asterisk asterisk asterisk" sixty times.



Diga! uses a database approach to store phone numbers, with associated scripts and configurations. Clicking on the words Script: or Config: (right of pointer) overlays a list of available choices.

Remote control

One of Diga!'s more interesting options is its ability to turn your Amiga into a poor-man's bulletin board system. With its Remote mode activated, you may call your terminal from another computer anywhere in the world and upload or download files. You may also list or change directories and view files' contents.

The Diga! remote allows two levels of security: "user-level" allows the basic access, while "maintenance-level" adds delete and copy capability. You may specify passwords for each level, as well as a file to display a welcome sign-on message.

While the Remote mode generally works well, some users have discovered idiosyncrasies. The first problem is more a function of the default modem setup included in Diga!: Configured for Hayes-compatible modems, it uses "ATS0 = 1" to tell the modem to auto-answer. Many modems, including the popular Avatex 1200, do not recognize the spaces. Change to "ATS0=1" for better luck. I have also found my modem works better when I send it a ATV1 and an ATX1, prior to entering the Remote.

The other Remote quirk is a matter of semantics. If you called up someone's Remote system and saw "S)end a file" and "R)eceive a file", which would you use to transfer a file to the other system? If you chose "S)end a file", you are wrong, but you're not alone. The Diga! Remote's send and receive are relative to the Host terminal, not the caller. If you want to send a file to the Diga! Remote, you want the Remote to receive it. Right? So to send, hit R)eceive. Argh. The Remote supports all Diga! transfer protocols except DoubleTalk.

Automating Diga!

Diga! incorporates a rich script language which allows users with a little bit of programming saavy to automate most major functions. Using a script, you can tell Diga! to wait until a low-access time (typically, the early morning hours), dial a selected information service until connected, collect waiting mail in a capture buffer and sign off.

Besides the now-standard script functions, such as conditional branching and file transfer instructions, the Diga! script language allows you to execute AmigaDos commands. In one example, the manual demonstrates how to tell Diga! to de-arc a downloaded file. Pretty nifty.

The manual does a fairly decent job of indoctrinating users in the proper syntax of script commands, with copious examples throughout. In addition, Diga! includes two pre-written scripts that will prove instructive in discovering the ins and outs of the language. No "learn" mode is included, so all scripts must be written from scratch.

Once you have a script up and running, Diga! offers several ways of executing it. You can select DO SCRIPT from the Project menu, automatically execute the script from the Phone Book, or click the script's icon from the Workbench. To execute several scripts sequentially, hold down the shift key, while clicking once on each script icon desired. A final double-click on the Diga! icon completes the sequence.

Gurus and gotchas

After using the program for only a short time, enough bugs become evident to make you wonder what the beta testers were doing with their time. In addition to Diga!'s problems with multitasking on 512K memory systems, and its occasional lockup when changing emulations, users have reported problems when moving between complex functions. For instance, problems surface when exiting the memory-overlaid Doubletalk, and immediately switching to an interlaced screen. Parity selections 7E1 and 7S1 are reversed on the pull-down menu, but they select correctly, using the Fast menu.

There does not appear to be a way to get Diga! to echo a line feed to your screen during a chat. Hitting RETURN returns the cursor to the left-most column, but it remains on the same line. Hitting CTRL-J moves the cursor down, but frankly, why should you have to? Finally, Diga! always demands its program disk be reinserted when you exit. If you ignore the requester, it's a guaranteed Guru.

Pass the Kelvar

Aegis customer support is always willing to help, and the company has even released a list of suggested work-arounds and undocumented features (See "Diga! Tricks".) The bottom line, though, is the software needs to be fixed. As I write this in mid-July, Diga! programmers Keary Griffin and Rocky Stargel are burning the midnight oil "bulletproofing" Diga! version 1.1, which Aegis hopes to have ready by the time you read this.

Besides debugging, Keary & Rocky are adding new features to 1.1. A tentative list includes WXmodem and Zmodem transfer protocols, and split-screen CHAT window capability. In addition, I have heard some tantalizing talk about beefing up the script language with new commands.

Aegis marketing VP John Skeel promises the upgrade path for owners of Diga! 1.0 will be extremely easy-- mail in the original, labelled disk and Aegis will send the upgraded version.

Overall

I had high hopes for Diga!; I still do. The program shows tremendous potential, and I find the more I work with it, the less problems I have. Nonetheless, even if one overlooks the bugs, Diga!'s documentation and human-engineering are surprisingly clumsy. To paraphrase one frustrated Diga! user, writing on one of the major information services," telecommunications on the Amiga simply shouldn't be this difficult. Back to the drawing board."

•AC•

Aegis Development, Inc.
2115 Pico Boulevard
Santa Monica, CA 90405
(213) 392-9972

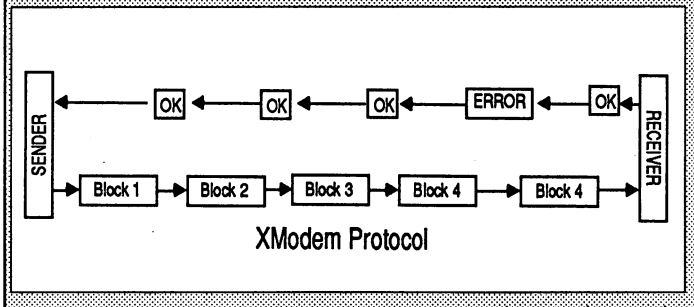
Special thanks to Deb Christensen, Henry Cisneros, and Jim Dlugokinski for technical support and testing assistance.

The DoubleTalk Protocol

To understand and appreciate the usefulness of a protocol like DoubleTalk, it helps to have a basic idea of how a conventional transfer protocol works.

The most common protocol today, XModem, sends data in 128 byte "blocks", one at a time. The receiving terminal checks each block, and if the block arrived intact, it signals the sending terminal that it's ready for the next block. If the block arrives garbled, the receiving terminal signals the error and the sender resends the block.

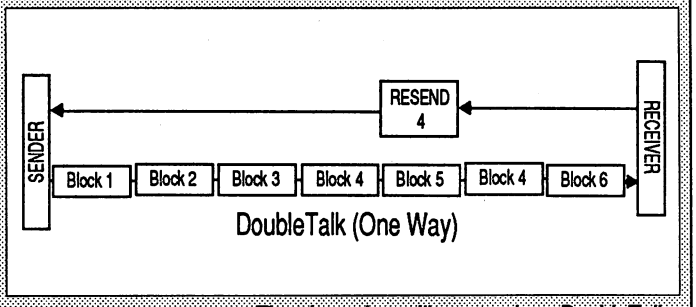
In the chart below, the example shows what happens when block 4 arrives garbled. This send/receive/acknowledge cycle is called handshaking and as long as the phone circuits don't induce a delay, it is an efficient way of transferring files.



XModem Protocol

A number of things can bog down XModem. Besides noisy phone lines (which will slow any protocol), packet switching delays in long-distance phone relays - especially satellite links - can also induce as much as a half-second of delay to signals travelling phone lines. This delay is not usually a problem when talking to a friend, but the extra half-second drastically slows the XModem protocol, which has to wait the extra delay before it sends each block. The cumulative delay adds over three minutes to the transfer time of a 400-block file - a time penalty of nearly 50 percent.

DoubleTalk gets around this problem by minimizing handshaking. After the DoubleTalk link is established, the sending terminal sends 128-byte blocks in a rapid stream, separated only by short inter-record gaps. If the block arrives intact, the receiving terminal places the data in its file buffer. If corrupt, the receiver sends a request to the sender to resend the block.



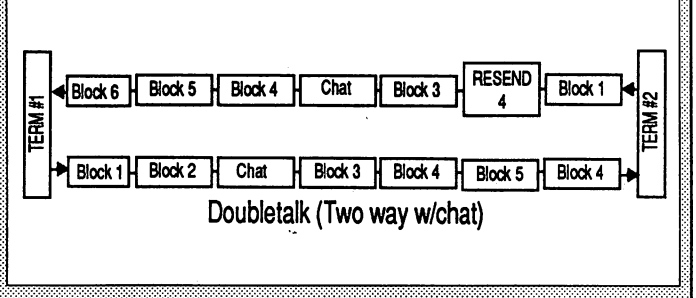
DoubleTalk (One Way)

The chart above illustrates how DoubleTalk would handle a garbled receive on block 4.

The question then arises: What would happen if the receiving terminal lost carrier (hung up) moments after establishing DoubleTalk? Wouldn't the sender keep sending blocks "into thin air?" To counter this quirk, the receiving terminal sends a handshake when it dumps its file buffer to disk, (about every 48 blocks.) If the sending terminal does not receive a handshake after a certain number of blocks, it times-out and the transfer is aborted.

Simultaneous sending and receiving isn't as tough as it sounds. As a matter of fact, the ability to send and receive a signal simultaneously is practically the textbook definition of a full-duplex communication line. At communication baud rates, most computers are twiddling their CPUs with more than enough extra cycles to handle two-way data transfer.

Because one side of the duplex transmission is not tied up with handshaking every other block, DoubleTalk is able to send another channel of data down that path. Therefore, data travels both ways simultaneously at full speed.



Doubletalk (Two way w/chat)

During a chat, DoubleTalk packs text entered at the keyboard into chat blocks, interrupting only at a carriage return to send the block.

-- Steve Hull

Diga! Tricks

In response to a barrage of calls to their Technical Support lines, Aegis has assembled a file of helps and hints for Diga!. Here are three of the most useful tips:

According to Aegis, Diga! was tested extensively, though the release version was set up differently than the test versions. Small differences - including the seemingly-innocent inclusion of a color-cycling "display" command in the startup sequence - have led to Guru system crashes that eluded the beta testers. To keep the Guru at bay, Aegis recommends you take these steps:

Delete the Cycles file from the Diga! root directory and the Display command from the Diga! C: directory.

Copy the commands Stack and BindDrivers from a Workbench 1.2 C: directory to the Diga! disk C: directory.

Using Ed or another text editor, modify the Startup-Sequence file in the Diga! S: directory as follows:

```
BindDrivers
Stack 10000
Type S/Startup-Sequence
LoadWB
EndCLI >nil:
```

•Finally, click on the Diga! Icon from the Workbench and select INFO from the Workbench menu. Change the STACK setting to 10000 and click on SAVE.

VT-100 emulation

To work correctly in VT-100 mode, the Diga! screen must be configured to 24 lines (overscan optional), with the cursor set to "transparent" on the Display menu.

The VT-100 programmable function (PF) keys may be accessed by using the Amiga numeric keypad, in conjunction with the CTRL key:

CTRL-9 = PF1
CTRL-6 = PF2
CTRL-3 = PF3
CTRL-. = PF4
SHIFT-. = ,

Saving memory

The Workbench screen may be toggled on and off by pressing CTRL-HELP. The Workbench screen must not have any active windows for this feature to work.

The complete Diga! Tricks file is in the public domain and may be found on most major information services.

-- Steve Hull

BILL VOLK

Vice-President, Technical Development
Aegis Development

by Steve Hull

People Link & Bix: ST.EPHEN
GEnie: LIGHTRAIDER

In an industry where machine loyalties spark Jihad-like behavior in otherwise rational people, Aegis Development's Bill Volk is an anomaly. He simply refuses to take sides, and it's not for lack of an informed opinion. Since 1979 he has made a living programming nearly every personal computer made, ranging from Radio Shack's TRS-80 Model 1 to the Commodore Amiga (One noteable exception: the Commodore 64).

In 1979 he received his undergraduate degree in Physics and Astronomy from the University of Pennsylvania. During his first year of graduate studies he signed on as a beta tester with Avalon Hill, where he quickly expressed his displeasure with the text-oriented ports of mainframe games. To prove his contention that graphics would boost sales, he developed the strategy-adventure Conflict 2500. Volk followed Conflict a year later with the 3-D real-time adventure Voyager I, a game Volk describes as a "pre-Wizardry Wizardry." Volk exhibited no small measure of wizardy himself, writing versions of the game for the Atari 800, Apple II, TRS-80 Models I and III, Radio Shack Color Computer, Commodore PET and the IBM-PC.

"I think in the long run people who have bought Amigas and people who have developed for the Amiga are going to be well ahead of the game."

In 1982 he began a three year business relationship with Epson/Rising Star. It was there that Volk wrote his first CAD (Computer Aided Drafting/Designing) package, for the ill-fated VALDOCS system. "They told me it had to run in 50K and use the disk as virtual memory," Volk says, "It was a real learning experience."

In 1985 Volk helped found Aegis Development, Incorporated. The young company's first products included Volk's Pyramid of Peril, a 3-D real-time adventure for the Macintosh, and Mac Challenger, a space shuttle simulator Aegis pulled off the shelves following the real-life Challenger disaster.

Aegis wasn't the only new face on the scene in 1985; that was the year the Commodore Amiga made its debut. Out of curiosity, Volk attended the first Amiga Developers' Conference — and from that moment, Aegis' direction took a profound change.

Aegis' products in that first year became dramatic testimonies to the power of the Amiga. Aegis Animator, a program offering real-time metamorphic and cel animation, was available within six months of the Amiga's launch. Volk's CAD package, Aegis Draw, was on the shelves before the end of the first year. And while all of that was happening, a little-known former Air Force pilot named Jim Sachs used Aegis Images to render a collection of sleek Porsches that have since found their way into the art collections of Amiga owners from San Francisco to Stockholm.

In Aegis' second year, the company has announced products that, if they perform as advertised, will move Aegis developers from the programmer category firmly into the magician realm. Telecomputing programs that can send and receive simultaneously? Graphics programs that animate complex 3-D solids in real-time? How did all of this come about? We decided to start at the beginning.

SH: How did Aegis decide to support the Amiga?

BV: How we got involved with the Amiga is a great story. I had been programming on the Atari 800 series for several years, and along around 1983 rumors began circulating that Atari was working on a 68000-based machine with a super graphics chip — which turned out to be, of course, the Amiga.

SH: But not produced by Atari.

BV: No, in a turn of events, Jack [Tramiel, founder and former Chief Executive Officer of Commodore Ltd - Ed.] went to Atari and the Amiga went to Commodore.

SH: Sounds like a fair trade to me.

BV: In March of 1985 we heard about two new machines coming out — the Commodore Amiga and the Atari ST. I called up both companies to see what I could do about getting hardware. Commodore told us that they would be holding a Developer's Conference in Monterey in May, suggested I attend, and they'd tell me everything about it — wonderful! Atari, on the other hand, basically asked me for $5,000.

SH: And so you went with the Amiga.

continued...

BV: We do ST products now and I think it's a good machine in its niche, but that's basically why we went with the Amiga at the time.

SH: How did the Developer's Conference go?

BV: The hall where they held the Conference was packed; there were a couple hundred people there, half from the Macintosh world and half from the Commodore 64/128 world. I had not seen the Amiga before and knew very little about it, other than the rumors I'd heard. The first thing that happens is, we see this giant projection TV — the Amiga's hooked up to it, and they run the "boing" demo.

SH: What was the reaction?

BV: People were literally in tears. And if that didn't get them, the "Smoke on the Water" guitar riff certainly did. It was amazing — people were really impressed. If you were to poll people as they left that show, I would say that about 90% of the people were totally committed to the Amiga. Absolutely committed. I'm talking companies like MacroMind, creators of VideoWorks on the Macintosh. Companies like Borland. And most of them never really did Amiga software.

SH: What changed their minds?

BV: What happened from that point is a story of commitment as opposed to just infatuation. A lot of people were scared off because of Commodore's financial problems. There were technical reasons, too. The Amiga is multitasking — and that was really a new age for some of the C-64 people who weren't used to dealing with the operating system. The most difficult thing about programming on the Amiga is managing resources; it's a big operating system and a lot of people aren't used to it. These people were used to writing routines to do everything themselves, and now suddenly they had to be very careful about how they allocated memory and things like that.

SH: The earlier versions of the operating system probably didn't help much.

BV: The Amiga was in such a state of flux in March of '85 — before the release of 1.0. If you talk to people from Island Graphics, they'll tell you it was really gruesome — but they started out before there even was an operating system, so they really had a tough time with it. By that summer when we began programming, it wasn't easy, but it wasn't that bad — everyone paints a different picture. We began working with the Amiga just before 1.0 was released.

SH: It sounds like Amiga developers had a pretty rocky first year.

BV: I guess it was like, "The few...the proud...the tough" that stuck it out. The companies that were committed — Electronic Arts, Aegis, and several start-ups — decided to stick with the Amiga, sacrifice some profits, put out software and see what happened. And it's paid off — paid off big for us, and I've heard that in terms of Electronic Arts' software sales, the Amiga is second only to the Apple II. When you consider the amount of Apple IIs that are out there compared to the number of Amigas, that's an amazing feat.

SH: By 1985 Electronic Arts was a large company with an established reputation. Didn't Aegis' commitment basically amount to a "make-or-break" gamble?

BV: More like "no risk — no gain." Aegis was a very small company before the Amiga came out. We had a couple Mac products, a little telecommunication, but it was basically a very small operation. There are really only two ways a software company can get started nowadays. One way is to get a tremendous amount of venture capital and hit the streets running — Electronic Arts and Lotus both started that way. The other way is to leverage off a new machine — one that may not otherwise be getting a lot of attention — and become a big company by supporting that new machine. It's risky, but I can't see any other way of doing it.

SH: Why didn't more larger companies take advantage of that "leverage"?

BV: It's one thing to take risks when you have five employees. It's a another story when you have 40. Once a company reaches a certain size, it's a lot harder to take risks — you have a lot of people depending on you. I'll give you example. We never thought we were going to do an animation system for the Amiga because Macromind, the people who did the Videoworks animation system for the Macintosh, were at the Conference. We thought for sure they would do it.

SH: Why didn't they?

BV: Two reasons. First, as a more mature company I don't think they wanted to take the risk. Second, the Amiga is so different from the Macintosh in some ways that they just couldn't "move over"; it was just too hard for them to take that learning curve.

SH: A tougher learning curve than the Mac?

BV: A different learning curve. The Mac probably has the most complicated operating system there is — even more complicated in many ways than the Amiga operating system. I know Jim Kent, the author of Animator, would disagree with me, but it's true. The Mac is much more complicated. The reason you see so many nice applications for the Macintosh is because the tools for developing on the Mac are first rate; for example, you can program all your menus, dialogs, and controls without actually writing a line of code. It's all done with a graphic-oriented Resource editor.

SH: How much of that complexity found its way into the Apple IIGS?

continued...

Amiga Programmers!

Put your features where your mouse is.

PowerWindows™

Version 2.0

*"Just wanted to let you know what a great development tool PowerWindows is. All the menus and requesters for our TV*TEXT program for the Amiga were produced with PowerWindows. PW has saved us countless hours of development time and is worth many times its price for that reason alone."*

-John W. Bittner, Jr.
President, Zuma Group, Inc.

***Power*Windows not only allows you to create custom windows, menus and gadgets for your programs at the drag of a mouse (complete with fully commented "C" or 68000 assembler source output for instant installation) it also provides these powerful new features in Version 2.0:**

- **edits multiple windows simultaneously**
- **creates string, integer, proportional and BOOLean gadgets**
- **reads in IFF brushes for gadget imagery**
- **allows easy menu positioning**
- **designs custom screens with extensive palette control**
- **includes menu mutual-exclude support**
- **generates optional event handler**
- **supports Modula-2**
- **lets you read in and edit any existing window with its unique "Grab A Window" function**

The experts use *PowerWindows*:

"PowerWindows is an essential tool for Amiga software development. It was used extensively in the development of Draw Plus."
-William Volk
Aegis Development

"PowerWindows gives you the freedom to be creative with the positioning of your menus and gadgets without the tediousness of coding and re-coding."
-Gary Bonham
Sparta, Inc.

"Excellent product. It saved me weeks worth of work. I generated all the windows, gadgets and menus for LexCheck with PowerWindows. ... I look forward to more great stuff from Inovatronics."
-Douglas King
COMPLETE DATA AUTOMATION, INC.

---

only $89.95

The REAL POWER in power programming. Period.

INOVATRONICS, Inc.

11311 Stemmons Freeway, Suite 8
Dallas, Texas 75229 (214) 241-9515

Available direct or from your local Amiga Software dealer.

Distributed in the U.S. by American Software, CSS/Apex, and Southern Technologies
Amiga is a trademark of Commodore Business Machines

BV: It's funny, but Apple has obviously looked at the Amiga; if you look at the operating system that's on the IIGS, you'll see a lot of Amiga-like things on it. The most apparent feature is the complete simplification of the operating system that makes it "look" like the Amiga operating system in the way it handles events. Very interesting story.

SH: We've digressed quite a bit from the Developer's Conference — what did you do when you got home?

BV: Following the conference, we had to decide what kind of software we were going to produce for the Amiga. I had previously written a CAD package for the Epson VALDOCS system that did very well — the CAD system, that is.

SH: What was your vision when you began working on your first Amiga CAD program, Aegis Draw?

BV: Let me say something right at the start; I'm normally the one who gets the credit for Aegis Draw, but getting it written was very much a team effort. Charlie Heath, for instance, is responsible for the file requesters throughout the program, and Jim Kent was invaluable in getting the IFF routines done. And there were others — everyone at Aegis contributed to the overall look and feel of the program.

When we began working on Draw, the premiere Macintosh draw program was MacDraw. Our goal in writing Aegis Draw was to do what MacDraw did, pretty much, but add floating point accuracy and some CAD functions like dimensioning.

SH: What exactly is the difference between a "draw" type program and a "paint" type program?

BV: They are two completely different ways of composing and storing graphics. "Draw" systems are object-based; a circle plotted using a "draw" package is stored in memory as a radius and a center — and perhaps a color and a pattern.

A "paint" package deals strictly with pixels. Because of this, a "paint" package can offer much richer quality, since it only has to deal with a pixel and doesn't have to do a lot with it. You have the capability to do a lot of colors, shading and effects that a real CAD package can't handle.

CAD applications on microcomputers have been slowly evolving. AutoCad for the IBM comes to mind right off the bat; I like to consider AutoCAD as the "Wordstar" of CAD products, and I mean that in a completely positive sense. Wordstar made word processing in the IBM computer world. The word processors that came before Wordstar looked like a typewriter; and face it, if you're used to doing your writing on a typewriter and you go to write a word processor, you're likely to use the analogy of a typewriter.

SH: That's the obvious metaphor.

BV: It's the obvious metaphor but not necessarily the best one. Today word processors have evolved beyond that to include pictures, multiple font sizes and so forth.

The point I'm getting at is, CAD products initially looked like drafting tables but since that time they have evolved. Since Draw Plus was released I've seen some new stuff in CAD that I'd really like to do if I have the memory to work with. Genlock is a good tool for doing drafting; you can shoot scenes or drawings with a video camera and simply trace them on the screen. It's actually a better way of doing it than if you had a sophisticated digitizing pad or other equipment.

SH: How does the Amiga rate as a CAD machine?

BV: The Amiga is a good machine for CAD, though not for the reasons most people think of at first. There are machines with more resolution than the Amiga. There are machines that are faster. There are machines that have better DOS's. But the Amiga comes in at a good price point with a good combination of features. It's stronger than its components; the Amiga is actually a very intricate machine. Electronic Arts got smart and pushed Commodore to establish a picture format, and that's made all the difference in the world. Amiga users take the IFF standard for granted but on other hardware it's not happening — it's not happening on the Apple IIGS, and it's not really happening on the Atari ST. The fact that IFF happened on the Amiga means that all the software that comes out from now on that has anything to do with pictorial information is going to accept or somehow generate IFF. This puts Amiga owners at a real advantage. You don't have to worry about whether your page-processing software will accept your CAD drawings.

SH: Do you find the Amiga's multitasking environment a help or a hindrance?

BV: Half the programmers we deal with love multitasking, the other half think it was a real mistake — but my feeling is that in the long run it will be a very significant move. Everyone's going to go to it anyway. In CAD it's a big deal because plotting with plotters can take a year and a day, and multitasking makes a big difference. Another benefit of multitasking with CAD becomes apparent when you hit SAVE and find out you don't have enough room on your disk. The Amiga will let you go back to the operating system while Draw is running and format some disks.

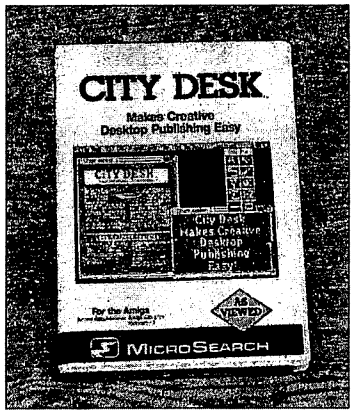
SH: Most Amiga users are still running with dot-matrix printers. Can you really do CAD with a dot-matrix?

continued...

Superior Products

"MicroSearch has a hands-on philosophy for product development. As a retail store and distributor, we constantly research the market. Our formula for success is simple. At MicroSearch, we listen to our customers...carefully."

We would like to thank Commodore for allowing us to exhibit with them at COMDEX Spring '87.



$149.95 (U.S.)

City Desk Sets New Standards in Desk Top Publishing

City Desk is a full featured Desktop Publishing Program designed with both the professional and amateur in mind. Now you have the power and flexibility to create high quality, professional looking documents. City Desk is for the serious users who demand the best, in products and results.

• Supports PostScript and HP LaserJet+

- 140 Page manual created with City Desk.
- Start Up exercise using text and graphics.
- Automatic kerning and leading.
- Powerful embedded command options.
- Unlimited font changes in the text.
- Flow text around graphics.
- Any number of fonts on a line.
- All preferences printers supported.
- Prints IFF pictures.
- Prints color pictures in gray scales.
- Text and graphic editors included.
- Not copy protected!

(This page was produced using City Desk and a PostScript printer.)

Head Coach Football Simulation

(Available September 1)

- Have instant replays, even slow motion.
- Show Stats while game is in progress.
- Player injuries and substitute players.
- Create realistic defensive alignments.
- Returns fumbles and interceptions.
- Call blocking assignments.
- Display jersey number or player strength.
- Computer can run both or neither teams.
- Create weather; wind, sun, rain or snow.
- Choose stadium type, name and surface.

If you've ever wished a computer football game could be more like a chess game... then its time you met the new Head Coach. Head Coach is a *strategic* game, not an arcade game. Playing Head Coach is as close to coaching the Pro's as you can get without signing a contract. You send in the plays, setting the strategy for those *exciting* long drives toward your opponent's end zone. You call the plays the same way a coach calls plays. It's easy and *fun* to have the QB hand off to the Halfback. And send him through the "three hole", by entering the simple command "RHB3". With Head Coach you can use the standard offensive or defensive playbooks or create your own. You can even design custom plays while the game is in progress!

$49.95 (U.S.)

Perfect Sound

A true stereo digitizer for your Amiga. Record any sound in mono or stereo, then use the Perfect Sound editor to modify the sound. Delete, insert, graph or flip recorded sounds.

$89.95 (U.S.)

Specifications

- Record both channels simultaneously.
- Lowest priced sound digitizer.
- Sample length: 8 bits.
- Sample rate (max): 23,283 per second.
- Frequency response: 11.6 KHz.
- Includes "C" source code.

Amazing Review...
"...This digitizer is fun and quite addictive! ...I highly recommend the PERFECT SOUND digitizer!"-- Ron Battle, Amazing Computing, Volume 2, #5.

$16.95 (U.S.)

EYE RESolution

The practical solution for virtually eliminating HI-RES screen flicker.

- Improves contrast.
- Attaches easily to the monitor.
- No messy tape or weak velcro.
- Simple "hook and hold" attachment.

MICROSEARCH
9896 Southwest Freeway, Houston, Texas 77074, USA
(713) 988-2818

BV: Well, the printer output on the Amiga is OK. Not good, not bad, just OK. One of the problems is that the Amiga has a lot of graphics capacity, and you want to be able to get that on paper. In Draw we did pretty much what everyone else did — we did a screen dump — which really doesn't look good at all. In Draw Plus we decided to solve the problem, but in doing so we ran into another problem — memory. To print a good quality picture, you need to internally create a very large display of what's going on. Apple did that with the Macintosh from the beginning, and it looks great — but of course, it's only black and white. On the Amiga if you want to get color high-resolution output, you need to set aside between 300 and 400K of display memory just for the printout — and that's what we did. If you've used Draw Plus with a dot-matrix printer, you see it's not bad.

Curious thing — the Draw Plus ability to change resolutions on the fly came about, not because we had planned it to do it that way, but because in order to get true hi-res printing on an 8-1/2 by 11-inch printout, you need to set up an 800 x 640 4-plane bitmap in memory — and the only way we could do it was to remove all our windows. Then I realized that once I got rid of all that stuff, I had to redefine it anyway. That came about very late in the game. Right up to final release, there were two icons on the Workbench for the different resolutions.

SH: What do people typically use their CAD products for?

BV: That's an interesting question. When we originally did Aegis Draw, we expected people to use it for simple illustration and planning-type drawings — for instance, how to put this table in this room. We found out that wasn't the case at all. People started using the package for things we had never intended, which was one of the reasons we decided to produce Draw Plus. People have used our CAD packages primarily as architects and technical illustrations. Robo City News and Amazing Computing have both used Draw for technical block diagrams.

SH: It sounds like you have made a sizeable commitment to desktop design, but Aegis is better known in the Amiga community for its desktop video products like Aegis Animator.

BV: As much as I have personally invested in CAD, I hate to say this — but desktop video is going to be bigger. I think — I really feel that the video market will be as big as the publishing market. I really do. It's an extremely visible marketplace; every company of any size does their own in-house videos, for training or whatever. There have been 'way more than a million camcorders sold, and those people with the 8mm or VHS decks didn't buy them just to watch movies. People are going out there and taping — you see them everywhere you go.

SH: And the Amiga is perfectly suited to home video.

BV: That's true, but that's almost an accident of history. Video resolutions had been plodding in hardware all the time, and the Amiga was being developed just about the time personal computer technology was approaching television-type resolution; basically 400 lines. Amiga made the decision to make an NTSC machine, which has its strengths and its weaknesses. The good thing is, you can do videos with the machine, and I think the video market is so huge, people haven't even realized how big it is. The bad news is interlace flicker, but even that will be fixed when Commodore introduces their high-persistence monitor.

SH: The demonstration tape of VideoScape 3D shown at the last San Francisco Commodore Show was breathtaking. I understand Allan Hastings broke some new ground with his 3-D rendering techniques.

BV: The VideoScape 3D package has helped us a lot. You see, there are basically two ways of doing 3-D. Packages like Easy 3-D or Mac 3-D are integer-based, which is to say they use whole numbers to store coordinates. And they're good — they're fast — but they aren't accurate enough to represent real-world objects as well as you'd like. The other 3-D method involves use of floating point math, which is accurate but the routines are painfully slow. Allan Hastings came up with a couple of nice tricks that made it possible to render three-dimensional solids using floating point math at a reasonable clip. And if all you need is wire-frame, it runs in real-time.

SH: Which opened the door to products like VideoScape.

BV: Not just VideoScape. It also relates to CAD and I'll tell you why. When we do 3-D passes in a new CAD package we're working on, we'll use Allan's three-dimensional graphics rendering routines. They're extremely accurate. You could draw a picture of the Earth and fly down and land on someone's doorstep.

SH: There have been some rumors that Aegis had found a way to extend the number of colors available in VideoScape animations.

BV: We use dithering to give you more than 16 or 32 colors; I think it really does 128 colors. It has its own color palette and allows true RGB color mixing; that is, two bits of red, two bits of green, two bits of blue and one bit of transparency. Those are the the colors you use, then what it does, is develop the in-between colors depending on where the light source is, and whether the material is matte or shiny. The dithering technique is similar to that used by Digiview for their hi-res pictures, which look as good or better than their strictly HAM digitizing.

SH: When VideoScape was first announced, Aegis said users would have to use a VCR capable of recording single frames to animate 3-D solids. Isn't that a pretty heavy limitation?

BV: Since that announcement in February we have developed what we consider to be a real breakthrough; that is, a system for recording, compressing and playing back IFF animation in real-time. This system, which we call Anim, has retargeted VideoScape 3D's user base so drastically we had to redo the documentation and even portions of the program's user interface. Before Anim, we felt the need for a single-frame VCR would limit VideoScape's users to a mostly professional video elite. That's all changed. Anim makes it possible to run several seconds of complex animation in real-time on a standard 512K system, so we had to go back and make the user interface a little friendlier for the average home user. Recording Anim still requires a megabyte of memory, because it has to compare two full screens in memory at once.

SH: How does Anim work?

BV: Anim works on the principle of differential compression; that is, when given two frames of animation to store, it only stores the change from frame #1 to frame #2. Not the whole frame — the differential. That's how it handles real-time animation. On top of that are the standard compression routines. Put this all together and an animation frame can be as small as 500 bytes in size; a 100 to 1 compression ratio.

SH: Is it possible to integrate screens created with Deluxe Paint or Aegis Images into an Anim file?

BV: The Anim format can be used to animate anything; it can use paint systems, draw systems — you can even use the CLI to create animation. We're working on a product called GrabAnim, which is something like the screen grabber, Grabbit, only instead of capturing a single screen, it'll store successive frames using the Anim format. It'll run in the background off a joystick or mouse in the #2 port; just click to begin recording single frames.

SH: Will other companies be using this format?

BV: I can't say that but I think it's pretty likely. We've released the standard, and it follows IFF conventions. We consider Anim to be a universal animation format because any package that does animation can easily generate it using GrabAnim. That, and the animation playback system is relatively small. You're going to see a lot of 3-D packages for the Amiga, particularly now that memory expansion is available at a reasonable cost.

SH: It seems funny to be talking about a half a megabyte of memory as a limitation when it wasn't that long ago that 64K computers were considered real power machines.

BV: That's one of the fallacies with the bigger machines. Would you be surprised if I told you the Commodore 64 has more RAM to work with than the standard Amiga?

SH: Surprised is not the word.

BV: I don't mean that literally, but follow my logic. A video display on the Commodore 64 can take up 8K of memory; that's exactly one eighth of the entire memory of the machine. On the Amiga with 512K, the video display we use with Draw Plus takes 128K of memory — that's one quarter of the memory in the whole machine gone before you've entered a line of code. Then too, you have to realize that everything a program does graphically — even the little things like the disk requester — takes up equally large amounts. On top of that you've got the 68000 processor — which is a great processor, but unlike the 6502, isn't exactly the most compact machine around.

SH: How much of a limitation is that in practical terms?

BV: We've got so much software "on the shelf" — we're just dying to have more memory to work with. Our character generator program, Video Titler, is a big program — it'll run on a 512K machine, but it has some features that don't become evident until you run with more than 2-1/2 megs!

Let me give you an example. Titler supports three types of text; regular Amiga text, the new Colorfonts system (Calligrapher), and its own "render-polygons" text mode, in which the letters are actually drawn rather than blitted out. Using those text modes, you can do about 1,000 effects. You can have it look like neon, you can have it look like light rays are bouncing off, you can make it look like it's chiseled in stone — and with polygon text you can also skew it and rotate it. Using its slideshow utility, Titler can integrate all this with the VideoScape Anim files. And it does all this on a 768 x 480 pixel overscanned screen.

SH: That does seem a bit much to ask of half a meg.

BV: The situation's turning around. I'm really excited about what's happening in hardware — the Amiga 500 is fantastic. And the great thing is that 1-megabyte 500's will soon be the standard Amiga configuration. I'd like to see the 500 become the next C-64. Even though the 2000 is a very interesting machine for developers, the 500 with a megabyte of RAM will be a reasonable machine for doing anything. I mean, you combine a 1-meg 500 with a reasonably low cost plotter, and you're talking about a $1200 CAD system. At that point, it costs about as much as a good drafting table.

SH: Aside from the price, what are your impressions of the new Amiga line? Some developers had hoped for further hardware enhancements, like higher resolution or more chip memory.

continued...

BV: The software compatibility is really good news. I know a lot of people were hoping Commodore would put new video chips in to make the machine really fancy, but by not doing so they've saved us a lot of trouble and saved the owners of 1000's having to worry about having an obsolete machine. The truth of the matter is, the new Amigas don't invalidate the old ones.

The people who want or need expanded capability can still get it. A lot of our customers have expanded their Amigas in ways that are very interesting. We have a bunch of people who are using the CSA Turbo-Amiga. This is the box that fits on the side of the Amiga and goes for the bargain price of around $4,500. It's got a 68020 in it and a 68881, and a bunch of RAM that runs really, really fast. It's faster than one of the best VAX'es you can buy. It's like having a super-minicomputer on your desk. One of our developers has one...it's really quite nice.

SH: Quite a range of capability between a $700 Amiga 500 and a $6,000 Turbo-Amiga system.

BV: It'll be hard to handle such a diversity of users with just one product so we're planning on offering several ranges of products. People who buy Amigas often buy them in lieu of much more expensive hardware because they know what it can do. I like to call the Amiga "the Sun workstation for the masses." It is an extremely competent system.

SH: Moving to one of your company's newer products, I get the impression that the public response to Digal has caught Aegis by surprise.

BV: It's been very good, actually.

SH: Really?

BV: We have sold thousands of Digal packages. By now, it may be one of the largest selling terminal programs for the Amiga. It basically comes down to this: people love the scripts — they're very powerful — they love the Phone Book, and they love DoubleTalk. They wish the program had more features and they wish the program was smaller. But the program is selling like hotcakes. Commodore ordered a couple thousand Digal's to sell themselves — Commodore already sells their own communications package.

SH: There were a couple of "gotchas" in Digal that I was surprised to see got by —

BV: You mean that it doesn't have WXmodem?

SH: Well, that's one —

BV: Do you want to know why it doesn't have WXmodem? Because the author of Comm told us last January that WXmodem was not yet a standard, and we should not support it yet.

SH: And People Link was running it.

BV: WXmodem was in the program right up to the second-to-the-last revision. We pulled it out because we were told there were two WXmodems and they weren't compatible. WXmodem is included in the upgrade we're working on right now.

SH: One of Digal's strong points is its expandability. Could you explain what influenced this design philosophy?

BV: Digal is a good example of a strategy we're beginning to take based on the extremely broad user base the Amiga has attracted. Historically, developers have tried to build "swiss army knives" — software with every option in the world, even if some of the features were so obscure that 90% of the users never used them. We've come to the realization that a package can't be all things to all people, and so you have to leave the software open — write software with "slots." Digal includes several of the major terminal emulations — VT-100, VT-52, and Tektronix 4014, and we have engineered the software in such a way that people may write their own custom terminal emulations and use them with the program.

SH: How much technical expertise will that take?

BV: Programming a terminal emulation isn't that hard and we've already posted the C source code for some of the emulations on many public-domain bulletin boards. Obviously, how difficult it will be depends on the emulation. Doing the Tektronics emulation would be very difficult.

It's like having a computer with expansion slots. Most end-users of the computer don't build hardware accessories for the expansion slots — they're not generally going out buying Protoboards and wire-wrapping their own stuff. However, the fact that the slots are there enables a whole second industry to start putting out low-cost multipurpose hardware — more hardware of different types than even Commodore is aware of. The same thing is true of Digal. The first add-ons we expect to see will be new terminal emulations.

SH: Since the beginning it has been Aegis' policy not to copy protect its software — in retrospect, are you happy with that decision or do you think it has cost you sales?

BV: There are times I wonder whether...well, it was a good decision. You can't sell productivity software that's copy protected. I can see protecting games as long as you offer a decent replacement policy. The thing is, on a multitasking system, copy protection is dumb. And people have problems with software that's copy protected. I hear there's even been some problems with copy protected software and the 500's, because the drives have changed.

I don't like dongles or other hardware approaches because dongles can be a madhouse — for instance, what if your dongle doesn't like my dongle? I do like documentation-based protection; one clever approach that you're seeing more often is the concept of actually having a quiz at the beginning of the game, asking for a word off one page of the user manual.

I don't know. I think software piracy's a serious problem with game software but I have to tell you one thing. There are lies, damned lies, and statistics, so let's take a statistic that I always hear: For every piece of software that is sold, two of them are copied. Well, there are two questions you should ask yourself. First, of those two people that copied that software - are they using it? Would they have purchased it if they couldn't copy it? I don't know. I can't just pick a number out, but I think the problem of piracy is much smaller than the actual number of copies that are out there. A lot of people copy just for the sake of having a program. What I don't like is people copying Draw Plus — because Draw Plus is so easy to use it's one of those cases where we may have boxed ourselves into corner. The package is so straightforward and self-documenting that it tells you what to do, and as a result people may have pirated it and gotten away with it. I don't think Digal or Videoscape will be very useful without the documentation.

SH: Is there anything else you'd like to say before we wrap up?

BV: I want to say something that's really near and dear to my heart, and that's the last year with the Amiga. You know, a lot of companies promised Amiga products back in that first year and didn't stick with it. They didn't stick with it because they read the press and the press said a lot of nasty things about Commodore. But I think in the long run people who have bought Amigas and people who have developed for the Amiga are going to be well ahead of the game.

•AC•

MouseTime and TimeSaver

Two Battery-Backed Clocks for the Amiga™

reviewed by John Foust

Although these two products are primarily advertised as battery-backed clocks, the Microbotics MouseTime and the C Ltd Timesaver are very different products.

MouseTime

The MouseTime is a metal box the same color and shape as Microbotics Starboard II memory board (Incidentally, these two products complement each other very well). The edges of the box are smooth and the corners are rounded. MouseTime fits quite snugly into the second mouse port. I had no worries that it would fall out. The device is the same height as the Amiga, so the case rests on the table top. Unlike some other mouse-port clocks, the MouseTime has a pass-through which allows you to use a joystick in the second port, if MouseTime is disabled. More on this feature later.

The MouseTime manual is very short; it fits on a single page. I can tell the developers struggled to make installation as painless as possible. Installation is always difficult for products that require changes in the startup-sequence file. If you install by hand, the Mousetime program means a one-line addition to the startup file.

The Workbench version of the date and time setting software is very easy to use. The setting system acts like a calculator (a much easier to use analogy than the date setting mechanism used in Preferences). A CLI version lets you set the system time to the MouseTime time, or load the system time into the MouseTime. This program should be invoked in your startup-sequence.

Software is provided to disable the MouseTime, thus enabling the mouse port pass-through. This event is a one-way event; that is, once disabled, the only way to re-enable MouseTimeis to turn off the Amiga or remove power from the MouseTime.

A spectacular crash of my Amiga disabled the MouseTime. I had files in my recoverable RAM disk, so power-down was not an option. Some danger of damaging the Paula custom chip did exist because the mouse port is very closely connected to the chip. I said "Paula, my dear chip, please forgive me," removed and re-inserted the Mousetime, and the clock was re-enabled. The manual does not recommend this procedure, but my Amiga is still alive.

MouseTime has the same quality feel as the Microbotics memory board. The hardware is nice, the software is nice - altogether, a very nice product. If you need a simple clock, MouseTime is certainly worth the $49.95 retail price.

Timesaver

Defining the C Ltd Timesaver as a battery-backed clock may be misleading. The TimeSaver has so many more features to overshadow the simple date and time function. So many more goodies, in fact, that I'd say you are paying for the macro abilities, rather than the battery-backed clock. This product says a lot for C Ltd. If they can put a full-blown microcontroller with K of RAM and ROM in this package and still charge only $80, they are certainly doing something right. This combination is a pleasant bait-and-switch.

There is little chance of Timesaver conflicting with any other device on your Amiga. Some devices use the parallel or serial port, and rarely have a pass-through to allow access to the port for other reasons. The Timesaver uses a port not used by any other device - the keyboard port. It is installed in-line on the keyboard cord. You remove the keyboard jack from the Amiga and plug it into the Timesaver, then plug the extension cord from Timesaver into your Amiga. The manual warns of grave damage if you confuse the two plugs. In the interests of science, I reversed the plugs with the power on. No damage resulted.

The manual recommends placing the Timesaver in a niche in the underside of the Amiga. A square of double-sided foam tape is included to hold it in place. I dread extracting my Amiga from its computer desk - it means I am forced to clean my desk - so I placed Timesaver on top of my Amiga, nestled against my memory expansion.

How does Timesaver set the time? As the Amiga boots, Timesaver squeezes the proper 'date' command into the keyboard buffer, then asks the Amiga to execute the startup-sequence as usual. If the boot disk has the 'date' command in the 'c' directory, Timesaver sets the correct time. Thus, Timesaver works without modifying any Workbench disks.

How does Timesaver perform its macro commands? Timesaver acts like a human typist. It sends commands to the Amiga only after it has intercepted and interpreted what you type at the keyboard. In this way, Timesaver can send several characters for a single keystroke. This ability is called a macro - the substitution of a single command for a series of commands (The word "macro" is used quite often in computers. It has the same general meaning. A spreadsheet macro executes a series of spreadsheet commands. A keyboard macro sends several keystrokes when invoked by a single keystroke.).

The Timesaver manual goes on and on; it should have been half the size it is. Of course, because of the extra macro features, the manual has a lot to describe, but sometimes goes too far. For example, the Timesaver packaging is a paper mailing tube. On page 63, the manual suggests you "can save more than time" by cutting a slot in the tube and using it as a piggy bank. I did not invent this one for an easy laugh.

Please, developers, hire a professional writer when the time comes to write your manual. I'm not just trying to keep my fellow writers alive. A well-written manual saves more than time. Strong documentation makes for a happier product owner. There are some very confusing passages, (such as the claim that 'dir opt a' is a built-in macro command of AmigaDOS that executes 'dir' on every directory in the current directory).

Some of the examples seem artificial. The tasks would often be better suited for a CLI 'execute' script. One example macro sets up the Amiga for compiling programs under Modula-2. It performs several 'copy' and 'assign' commands. In this case, I do not know why someone would not want to use an 'execute' script. If you had such a repetitive task, you could use the Timesaver to record your keystrokes, then play the

continued...

keystrokes back in your editor and save the file as an 'execute' script. I am sure Timesaver can do many similar tricks.

The most practical uses of Timesaver are the command line editing and recall of previous commands. Many Timesaver commands are invoked using the Help key as a shift key. Help-up-arrow recalls the previous line entered. The right and left arrow keys let you edit your mistakes or modify a previous line.

I don't trust most public domain CLI shells, even though they offer similar command line editing. Timesaver is insulated from the vagaries of AmigaDOS, so it is compatible with nearly everything. If Timesaver is not compatible, you can turn off the extra features.

The command recall works between CLI windows as well. This convenience would be very difficult to accomplish in a software shell. I often flip between screens. I noticed that the command recall recorded the Amiga M and N sequence as 'mn', as if I typed it in the CLI, so my command history was full of lines of 'mmnmnmnmnmnmn.'

So, is Timesaver strictly a tool for the CLI user. I don't think so. It is nice to have the same set of keystrokes for typing your name or address, for instance. This sequence could be used online in a communications program, word processor or spreadsheet - - - all with the same keystroke.

It is very possible to buy the Timesaver for the clock function, and then be seduced by the macros. There are many other features, too. Timesaver has a password protection option that won't send any keystrokes until the proper password is entered. A macro can be chosen to be executed on startup and can record mouse movements, if you use the keyboard equivalents for mouse movements. These features eat up much Timesaver memory, though.

Although seven thousand bytes are free for macro storage, it is difficult to estimate how many bytes of memory will be used by a given macro. The Timesaver must store the actual keystrokes sent to the Amiga, which include separate sequences for key-up and key-down.

C Ltd. is planning additional software to complement the Timesaver, including a program to load and save macro sets. This software will be distributed in the public domain. A different version of Timesaver will be available for the Amiga 2000. The Timesaver does not work on the Amiga 500 because the 500 does not have a separate keyboard. CLtd. is also planning a version that would allow you to use an Amiga 2000 keyboard on the Amiga 1000. The Timesaver design also serves as a translator for any IBM-style keyboard used on the Amiga 2000.

As you can see, the Timesaver has many more features than the MouseTime. I used the Timesaver and MouseTime for several weeks and both functioned flawlessly. In the course of writing this review, I thought of several new uses for Timesaver's macro commands. It is a neat hack.

In summary, these two clocks are so inexpensive and close in price, it is hard to imagine not choosing the Timesaver, if you can appreciate its extra features. If you want a simple clock at a low price, the MouseTime is a sound decision.

•AC•

Microbotics, Inc.
811 Alpha Drive
Suite 335
Richardson, TX 75081
(214) 437-5330

C Ltd.
723 East Skinner
Wichita, KS 67211
(316) 267-6321

Michigan Software's Insider Memory Expansion Card

reviewed by James O'Keane

As most heavy Amiga users know, an Amiga with only 512 K of RAM leaves a lot to be desired. Many RAM expansion products have appeared in the Amiga marketplace in recent months, all very seriously competing for the Amiga owner's dollars.

In past issues of Amazing Computing, several types of memory upgrades have been discussed. Most of these are of the external-box type, which is attached to the side of your computer. Larger, more expensive versions of these external devices allow multiple expansion cards in one chassis and/or their own power supply in the box. Most of these devices cost half as much as I payed for my Amiga 1000.

As a low cost alternative, many people are attracted to the do-it-yourself upgrade as described in the article in Amazing Computing by Chris Erving. This route is inexpensive - less than $100 - and an effective way to double your available memory, but it does have some disadvantages. It is a time consuming project and definitely is not for those of us with little soldering experience. What does that leave for those of us with money and desk space constrants? The Insider by Michigan Software.

The Insider is one of several currently available internal memory expansion products. It is fully assembled, some semi-kit products. The Insider is a nicely designed, 1 megabyte memory expansion board with a battery-backed clock. It must be installed INSIDE the Amiga, thus voiding any warranty remaining on your computer. Unfortunately, although much easier than the hardware hacker's special, the Insider is not for people afraid to roll up their sleeves and apply a screwdriver to the delicate insides of a computer. I don't mean to sound discouraging, but this product is NOT for the neophyte.

What You Get

The Insider arrived with a clear, concise manual, a floppy with several utility programs and the Insider PC board. The twenty-page manual is clearly written and deals mainly with the installation procedure for the Insider. The manual includes a support number to call if you happen to get stuck at any point during the installation. The manual also has many important warnings and notes about sensitive steps in the installation. These warnings are given before the step where the problem can occur. The manual also contains a page of very helpful diagrams showing the placement of jumper wires. A good quality manual for a product like this is ABSOLUTELY essential.

The included disk has the bare necessities of memory expansion utilities. AddMem is a program used to let the Amiga know that you have more memory than it decided on power-up. This program is only needed under AmigaDOS 1.1, or if you change the default starting memory location of the memory board. RTClock sets or reads the time and date from the battery-backed clock. MemTest tests the integrity of your new memory. Lastly, the program Ram On\Off disables and enables the Insider RAM. The program can be run from an icon on the Workbench.

The Insider hardware is in the form of an L-shaped PC board. Two rows of RAM chips, a clock chip, the address decoding chips and a 68000 socket are on the PC board. Another extension 68000 socket is used to place the Insider board at the correct height above the Amiga PC board. There are also two short wires terminated in hypodermic-type spring clips. These allow thelinsider to access two signals present on the daughter board. (The daughter board holds the writable control store, also known as the KickStart RAM). Attaching one of these two clips turned out to be the most difficult part of the installation.

Installation

I must admit the installation of the Insider was a bit trickier than I expected. Please note that I didn't say difficult. One of the steps is just not as easy as it may seem at first glance. The step-by-step instructions for disassembling the Amiga are not very difficult. I had my Amiga in pieces after about 15 minutes. Removal of the 68000 was a bit harder; a chip puller would have been a great help at this point (I didn't have one handy and made do with a small screwdriver). Radio Shack stores sell an IC removal and insertion tool set for a reasonable price-- if you plan to do this installation I would recommend buying one.

The installation of the Insider, as described in the manual, is very complete. My only problem came when it was time to attach the red jumper clip BENEATH the daughter board. I found this step particularly frustrating. I also managed to bend a few pins on the 68000 while attempting to push it into the socket on the Insider - nothing that couldn't be fixed with a bit of work with a pliers. Fortunately, Michigan Software sells replacement 68000 chips for $12.95. All in all, it was much easier than soldering chips on top of chips, but still not a job for the faint of heart!

The starting memory address of the Insider board is DIP switch selectable to one of nine address blocks. Only the $C00000 block will autoconfigure under AmigaDOS 1.2. If you have another hardware device that maps into this area, such as the memory in the PAL Jr., you will need to set the DIP switches for a different block of memory.

Compatability

As far as my tests have gone, the Insider works perfectly with the software I own. I downloaded the ASDG recoverable RAM disk software from a local BBS and it certainly works well! This software and memory expansion are a must for anyone who does a large amount of programming on the Amiga. Several older games will only work if I boot with the KickStart AmigaDOS 1.1 disk. Under 1.1, this memory is not recognized without the AddMem program.

Advantages

The Insider has many advantages over other memory expansion products available for the Amiga. It is available from discount mail order houses for less than $300. It has a one year warranty. It does not tie up the expansion port. According to Michigan Software, it works with the SideCar.

Disadvantages

The Insider, like most products, does have its flaws. The battery for the clock is non-replaceable. The advertisements say it has a ten year lifespan. And installation of the Insider is not as simple as your average hardware expansion product.

All-in-all I am satisfied with the performance of this product and I look forward to other low-cost hardware from Michigan Software.

•AC•

The Microbotics Starboard-2

by Steve Faiwiszewski

While 512K in your Amiga may seem more than enough at the beginning, there are instances when more memory is necessary. I've reached a point where more RAM became a necessity, so off I went looking for a decent RAM expansion. What I ended up with was the Starboard2 from Microbotics.

Look and Feel

The Starboard2 is a small oblong metal box which fits nicely at the side of the Amiga. The color of the case matches the Amiga's color, making it quite aesthetically pleasing. The unit is the same height as the Amiga, and has a pretty small footprint (1.6"Wx4.3"Hx10.2"L). The case sticks out about an inch beyond the back of the Amiga, but I did not find this fact bothersome. However, the front of the unit comes to within half an inch of the second mouse port. This provides no difficulty in using the second port with a joystick, but it may interfere with other peripherals (such as Byte by Byte's Tic). On a more technical note, the Starboard has the following attractive features: 1) The unit has bus pass-through. This is very important to anyone who, like me, would never get a 'dead-end' peripheral, one without bus pass-through. Having a peripheral with pass-through assures me that I'll be able to expand the Amiga further without any problems. 2) the RAM is zero wait-state. This means that the CPU or the custom chips never have to wait for the RAM to "catch up". An amiga with zero wait state RAM expansion will run faster than an Amiga with RAM expansion that has wait-states. 3) The unit is a professionally engineered product; it fits onto the expansion port snugly and accurately. The solid case is supported by rubber feet, and does not put undue strain on the expansion port. Taking apart the case, one can see a cleanly designed printed-circuit board with no visible wire traces; a sign of good clean design.

When fully populated (meaning it has 2 Meg), the unit draws about half of the 1 Amp, 5 Volt power supply the Amiga puts out on the bus.

Hardware Configuration

The Starboard2 is made up of two printed circuit boards. The first board (known as the main deck) contains sockets for 1 Meg of RAM chips, as well as connectors for the second board and the multifunction board (more about that one in a minute). The second printed circuit board (called the upper deck) is needed only if you want to have 2 Megs of RAM, and it plugs into the main deck. Each deck also has sockets for parity RAM chips (4 on each deck). These chips are only necessary if you get the multifunction board and want to use the board's parity checking feature.

Ordering

The unit can be ordered in various configurations: with 1 or two decks, with 0K, 512K, 1Meg, or 2Meg worth of RAM chips. Why would anyone get a RAM board with no RAM on it, you ask? Well, theoretically you should be able to buy the RAM chips yourself at a much cheaper price. I for example ordered a Starboard with 2 decks but only 1 Meg (the upper deck is not populated with chips). However, I now regret that decision: at the time of this writing all this talk about the 100% tariff on Japanese imports caused the price of RAM chips to increase significantly.

When deciding on what configuration to order you should keep in mind that buying your own RAM chips may not be as cheap as it used to be. Also, Microbotics approves of only a few RAM chips (Texas Instruments, Hitachi, and Mitsubishi). Installing any other RAM chips would void the warranty. Iif you are thinking about buying the upper deck only later, separately from your main deck purchase, be aware that you'll probably end up paying about twice as much for the upper deck than if you'd buy the whole package together! Why? Because for some unknown reason (to me anyway) Microbotics charges the dealers more for unbundled upper decks. So my recommendation is, buy the full 2 Megs; I don't think you'll regret it. I've seen the 2 Meg configuration sold at Abel Supply (Voice # (615) 428-5100. BBS # (615) 453-0643) for $430.

What I received

The unit came in a plainly marked cardboard box, tightly packed in anti-static foam wrapping. Aside from the Starboard2 itself, the box contains the screws used to attach the unit to the Amiga, a disk, and a double sided sheet of paper with installation instructions. My dealer also supplied me a pair of small screws used to hold the case together. He maintains that the original screws supplied by Microbotics are of low quality and may wear out. Those screws are removed whenever more RAM (or the multi-function board) is added to the unit.

I found the instructions quite clear and easy to follow. They cover how to hook up the Starboard to the Amiga, as well as how to add more RAM. Make sure to read everything carefully! Don't assume that you know what to do. I initially ignored the instructions, and I almost damaged the main deck, because I didn't read that I shouldn't press on the case of the Starboard when attaching it to the expansion board (doing that flexes the main deck, and can cause it to crack). Luckily everything worked out fine.

Supplied Software

The supplied disk contains a RAM test program as well as AddMem for Kick/Dos 1.1. You are told to run the RAM test after you install the Starboard. Running the test is quite simple: after loading Kickstart 1.1 (must be 1.1; the test program won't run under 1.2) just insert the disk in DF0:, and the test will run automatically. Testing takes about 5 minutes, depending on how much RAM you have (the more RAM, the longer it takes). There is a 'read-me' file on the disk describing how to set up your 1.1 Workbench disk to enable all that extra RAM. That basically involves copying the AddMem and another program to the c: directory, and modifying s:Startup-Sequence to run AddMem. Naturally, no software is required for Kick/Dos 1.2, as the RAM auto-configures.

Benchmarks

Since all zero wait-state FAST RAM expansion boards operate at the same speed there is not much point in benchmarking the various RAM boards. If you're still curious, you can use the RamSpeed program written by Perry Kivolowitz which can be found on one of the Fish disks, and on CompuServe.

The Multifunction Board

Microbotics will come out with an additional board (you guessed it: the multifunction board) which plugs into the main deck and provides the following features:

• Battery-backed real-time clock/calendar. We all know how this can make life easier.

• Socket for the 68881 Floating Point coprocessor Unit. Anybody who uses the Amiga for heavy number crunching will greatly appreciate having an FPU. Unfortunately, the 68881, which must bought separately, is quite expensive at this time.

• Parity checking. The more RAM you have, the greater the probability of random errors will occur. The Parity checking feature will reduce the chance (for FAST RAM only) of such an error going undetected. When a parity error is detected, the Amiga will crash with a GURU message. Theoretically, crashing is better than letting the user save some data which has been corrupted by RAM errors.

• Hardware recoverable RAM disk. This will allow you to turn your FAST RAM into a RAM disk which will survive reboots. To be honest, now with the existence of such software products as the ASDG recoverable RAM disk, I can't see the need for hardware recoverable RAM disk.

Last I heard Microbotics will start shipping the multifunction board on May 15.

Possible Problems

I've heard some rumors that the Starboard may suffer from the same problem afflicting the C Ltd RAM board. The C Ltd board on some Amigas does not work well with other peripherals. C Ltd's representatives claim that the problem stems from some PAL chips in the Amiga that are of marginal quality. I've heard someone else say that the problem stems from the fact that the C Ltd board does not adhere to the Zorro standard (the connection to the Amiga bus is not electrically buffered).

Now, the Starboard too does not adhere to the Zorro standard, so there may be some truth to the rumor. I also heard that Microbotics is working on another peripheral which will have a power supply large enough to run 4 Starboard2's off it, and it will also buffer the Amiga bus from the Starboards. So far, though, my experience with the Starboard has been perfectly satisfactory. But then again, I don't own any other peripherals...

Another problem with having the Starboard is common to all FAST RAM expansion boards. Some programmers made the mistake of including graphics data in their software without specifying that this data should be loaded into CHIP RAM. Unless told otherwise, AmigaDOS tries to load the whole program into FAST RAM, where the data is inaccessible to the Amiga custom chips. The result is that any graphics output those 'misbehaved' programs have will appear distorted (or not appear at all!). Fortunately there are two solutions: a) Run the public domain program 'FixHunk' on the misbehaved program. This will cause any 'data hunks' in the file to be loaded into CHIP RAM. b) Before running the program, disable or 'grab' all the FAST RAM, so AmigaDOS won't have a choice but to load the program into CHIP RAM. ASDG's recoverable RAM disk software comes with a utility which will mark all FAST RAM as used.

Useful PD/Shareware Software

If you get the Starboard, or any other RAM expansion, be sure to obtain ASDG's recoverable RAM disk software. This shareware product is available practically on any computer system (CompuServe, PLink, GEnie, etc) and most Amiga-related BBS. The software sets up a ram disk called VD0: which behaves just like RAM:, except that if you reboot, or if the Amiga crashes, VD0: won't go away, and its contents won't disappear.

Another program called MakeACV (and a sister program called LoadACV) is PD, and can be found on CompuServe. These utilities are useful if you have a relatively large number of files you want to copy to RAM: or VD0:. AmigaDOS is slow at copying many little files, but it does okay at copying one large file. MakeACV will let you compress many files into one big file. LoadACV does the opposite: it 'unpacks' the archive into the component files. The time saved in moving files to RAM in this manner can be significant.

•AC•

ATTENTION! DELUXE MUSIC CONSTRUCTION SET™ USERS

ATTENTION! MUSIC STUDIO™ USERS

SYMPHONY SONGS

ATTENTION! INSTANT MUSIC™ USERS

SYMPHONY SONGS gives you a library of nearly 1,000 music masterpieces. All songs are in IFF format so they may be loaded, played, printed, transposed, and modified in any way you like using your favorite composition program. Included is a free program to convert the IFF files to MUSIC STUDIO™ format.

The songs have been arranged by C. Clark Rulaford and Randy Spector and take advantage of the full **4 Voice** capability of the AMIGA.

Space does not allow listing all the songs in each of the volumes. We have listed a few and show the total number in each volume as well as the playing time. A complete list of songs may be purchased for **$3.95.** Each volume of the 27 volumes listed is **$24.95.**

BEATLES Part 1
Vol 15 (21 Pieces 40 Min)
Let It Be, Yesterday, Eleanor Rigby, When I'm 64, . . .

BEATLES Part 2
Vol 40 (17 Pieces 40 Min)
Magical Mystery Tour, Lucy In The Sky With Diamonds, Penny Lane, . . .

CLASSICAL Part 1
Vol 27 (19 Pieces 40 Min)
Prelude #1, Moonlight Sonata 1st and 2nd Movement, . . .

CLASSICAL Part 2
Vol 34 (15 Pieces 40 Min)
Sonata In C Major, Jesus Joy Of Man's Desire, . . .

CLASSICAL Part 3
Vol 31 (18 Pieces 35 Min)
1st Piano Concerto, Polonaise Sonata In C Major, Etude #3, . . .

CLASSICAL Part 4 (Bach)
Vol 35 (22 pieces 30 Min)
Two Part Invention #1, Three Part Invention #6, Prelude and Fugue 1, . . .

CLASSICAL Part 5 (Bach/Clementi)
Vol 46 (24 Pieces 50 Min)
Choral #1, Sonata #1, Theme and 11 Variations From The 2nd Sonata, . . .

BEETHOVEN, BROADWAY, & BLUES
Vol 38 (15 Pieces 40 Min)
2nd Movement Of the Pathetique Sonata, Minuet In G, Fuer Elise,

COUNTRY CLASSICS Part 1
Vol 41 (18 Pieces 45 Min)
Thank God I'm a Country Boy, Act Naturally, . . .

ROCK Part 1
Vol 32 (19 Pieces 50 Min)
AXEL F, Eye Of The Tiger, Both Sides Now, . . .

ROCK Part 2
Vol 16 (21 Pieces 40 Min)
Georgy Girl, Guantanamera, Theme From "Love Story," Cherish, . . .

80's GREATEST
Vol 24 (16 Pieces 50 Min)
Hill Street Blues Theme, Chariots Of Fire Theme, Dynasty Theme, . . .

70's GREATEST
Vol 12 (21 Pieces 45 Min)
Tie A Yellow Ribbon On The Old Oak Tree, We've Only Just Begun, . . .

60's GREATEST
Vol 13 (21 Pieces 45 Min)
Windy, By The Time I Get To Phoenix, Come Saturday Morning, . . .

GOLD & PLATINUM HITS
Vol 45 (19 Pieces 60 Min)
Thriller, 99 Luft Balloons, California Girls,

KENNY RODGERS HITS
Vol 39 (12 Pieces 45 Min)
Lady, Ruby, She Believes In Me, The Gambler, . . .

BILLY JOEL GREATEST HITS
Vol 43 (17 Pieces 65 Min)
Piano Man, Say Goodbye To Hollywood, Only The Good Die Young, . . .

COUNTRY CLASSICS Part 2
Vol 42 (19 Pieces 50 Min)
Ode To Billy Joe, Me and Bobby McGee, Country Roads, . . .

TV THEMES
Vol 37 (21 Pieces 35 Min)
Hill Street Blues, St. Elsewhere Theme, Masterpiece Theater Theme, . . .

MOVIE THEMES
Vol 19 (23 Pieces 40 Min)
MASH Theme, The Rose, Can You Read My Mind (Superman), . . .

BROADWAY'S THEMES
Vol 47 (25 Pieces 65 Min)
The Last Supper, Dr. Doolittle, The Old Dope Peddler, . . .

CHURCH MUSIC
Vol 28 (26 Piece 50 Min)
Amazing Grace, What A Friend We Have In Jesus, . . .

BARBERSHOP
Vol 22 (22 Pieces 45 Min)
Hello Dolly, Put On a Happy Face, Hey Look Me Over, . . .

RICHARD RODGERS SONGBOOK
Vol 18 (19 Pieces 40 Min)
Climb Every Mountain, DO-RE-MI, The Sound Of Music, . . .

NOSTALGIA
Vol 17 (22 Pieces 45 Min)
Let Me Call You Sweetheart, Ain't Misbehavin', On The Goodship Lollipop, . . .

CHRISTMAS
Vol 36 (26 pieces 50 Min)
O Little Town Of Bethlehem, Let It Snow, March Of The Toys, . . .

POLKA PARTY
Vol 33 (18 Pieces 40 Min)
Happy Polka, Pizzacato Polka, Betty Polka, . . .

SYMPHONY JUKEBOX

Symphony Jukebox allows you to program a selection of songs, their order as well as the number of times, and allows you to listen to them for hours of uninterrupted playing. Other features include MIDI output, instrument selection, transposition, and tempo modification. **$24.95**

SYMPHONY MUSIC VIDEO

This program has all the features of our SYMPHONY JUKEBOX, however, it also allows you to specify a picture to be displayed with each song. The pictures and music are all in standard IFF format so you may use the songs and pictures included, or use those you developed with your music program (i.e. DMCS), or your paint program (i.e. Deluxe Paint). Included are Christmas music and pictures. **$24.95**

We accept CASH, CHECK, COD, VISA and MASTER CARD orders.
Shipping and handling US and Canada $3.00
Shipping and handling outside the US and Canada $5.00
COD charge $2.00
Illinois residents add 6¼% sales tax.

VISA MasterCard
Dealer Inquires Invited

Speech Systems
38W255 DEERPATH ROAD
BATAVIA, ILLINOIS 60510
(312) 879-6811

AmigaNotes

by Rick Rae

CompuServe 76703,4253

This issue should be hitting the streets sometime around September. What with the unavoidable "back to school" flavor of this time of year, I thought it might be appropriate to take a look at a tutorial package. What I ended up reviewing was less, and more, than I expected.

Music Student I

Associated Computer Services offers quite a bit of unique Amiga software ranging from several volumes of clip art, through professionally oriented television graphics systems, to scholastic packages. Music Student I is one part of a multi-volume set; Music Student II is currently in the works and will pick up where this package leaves off.

I really expected Music Student to be a computer aided instruction package, especially considering the wording of the advertisement which mentions 178 "lessons" on the disk. What it is, in reality, is a set of computer administered quizzes for a music curriculum.

Music Student I is shipped in a shrink-wrapped 7x9 notebook, and includes one disk, 40 pages of instructions, a student score card, and a warranty registration card.

Music Student can be started by inserting the disk at the WorkBench prompt, or by running the program "Giver" from the CLI. The system is geared for a classroom environment rather than for individual use. When initially started it is in "instructor mode" where all options, including exit to the operating system, are available. Once the "Start Giver" option is selected, the system is in "student mode" and cannot be exited without knowing the password or performing a reboot.

Using Set Options, the instructor can modify some aspects of program operation, such as the number of tries allowed before Music Student provides the correct answer, or the pathname for the quiz scores file.

In "student mode" the introductory screen is displayed and the user is asked for his or her name, then given the "Select a Quiz" display. Music Student I has eight levels of difficulty, with each level containing from 20 to 24 quizzes. The level is selected with numbered gadgets down the left side of the screen, then the desired quiz is picked from the scrollable list displayed in the center of the screen. Clicking GO will start the desired quiz or, if no selection was made, will return the system to the introductory screen ready for the next student.

The questions cover a fairly wide range of knowledge, from "Describe the sounds on the right side of the keyboard (HIGH or LOW)" to "Name the scale that begins on the mediant of a major scale and uses the same key signature as the tonic-major". Most of the questions in difficulty level one are suitable for young children receiving their first musical instruction, so Music Student really does start at the beginning.

When a quiz is completed, the student's performance is recorded in the scores file and the Quiz Statistics screen is displayed. This display shows the number of correct and incorrect answers, along with a "hit" percentage and a comment which ranges from "Better review the material!" to "That's Perfect!". At this point the student is given the option of taking another quiz; if he chooses not to, the system returns to the introductory screen for the next user.

Bugs and Gnats... er, Nits

Although Music Student performs the job for which it was designed, it has several rough edges and a few outright bugs.

The most obvious problem to me, because I have an Insider board and 1.5 megabytes of RAM, is that this program will not run properly with expanded memory. The program's designer used a custom "checkmark" in the Giver menu and a square "Wait" pointer, neither of which will display properly if the program loads into Fast RAM. Users with more than 512K of memory should toggle their extra RAM off before loading this program. (It might also be possible to run a program such as FixHunk on the Giver program, but if you try this, be sure to do it on a backup copy, NOT the original.) Fortunately this is not a fatal bug, and the program is still usable even if loaded into Fast memory — but I am looking forward to the day when programmers realize that some people have more than 512K in their systems.

Problems with graphics go beyond this, however. If you run Giver and the Music Student disk is not the logged drive, many of the gadgets never appear! They are still functional, but their imagery simply doesn't show up. Nor do the graphics for the introductory screen, nor the graphics which are supposed to accompany the questions.

I found I could crash the system fairly consistently by specifying scores be saved to a second disk which was write protected, then selecting "Cancel" when the requester came up.

Most of these bugs won't bite the typical user: the teacher with a 512K Amiga who reboots the computer to load every program. But that teacher is going to have a task unique to the scholastic environment: deciding how to integrate Music Student into the classroom. This will be more difficult than it should be due to lack of documentation.

Music Student is, according to the manual, a "set of questions and answers for the study and review of music fundamentals". Fine, but how do you apply the program without knowing what material is covered? How can you know when to assign which quiz to what students, without knowing what the questions are? The only information provided in the manual is a "directory" which lists the general categories of quizzes, and this is of little help. The only way to determine what questions are going to be asked is to sit down and take the quizzes, something the instructor shouldn't have to do. At the very least there should be a list of all the questions and correct answers grouped by quiz; better would be a brief discussion of the material to insure the teacher covers it properly and uses terms consistent with the quiz.

Music Student is a good concept, for what it is: an aid to determining how well a pupil is learning and retaining material covered in class. But an automated quizzing program

continued...

is only the smallest part of teaching, and I found the required support missing. The program does what it is supposed to do and it does it reasonably well, but a bit more time spent polishing the code and the manual would have made a big difference.

Throughout the manual there are references to Quiz Master, and in fact Music Student was created with this program. So after finishing with Music Student, I took a look at Quiz Master — and found a whole new world.

Quiz Master

This program is intended for teachers who want to design their own computer aided quizzes. Quiz Master is versatile enough to incorporate IFF graphics and limited sounds in a test, yet is fairly simple to operate. Although there is no limit to the subjects which could be quizzed, all the tests will use Giver's standard interface, resulting in a degree of uniformity that students will probably appreciate.

Quiz Master is packaged identically to Music Student, almost down to the number of pages in the manual. The disk includes the programs Maker (the quiz editor) and Giver (the quiz administrator, also used by Music Student), and a set of example quizzes and tutorials. The Spatial Relationships quiz is an excellent, if brief, example of what can be done with Quiz Master; likewise, the States tutorial shows how Quiz Master can be both a tutor and a tester.

The quiz editor is started either by clicking on the appropriate icon or invoking Maker from the CLI. The black opening screen has two menu options: Project and Print. The latter option is similar to its namesake in Music Student but is much more powerful. In addition to student scores, this menu also allows you to print a directory of quizzes, and you can print to the screen, printer, or a disk file.

You can also print the questions from any quiz in one of three formats. They can be printed by themselves, accompanied by blanks and followed with an answer sheet, for administering a Quiz Master test by hand. Or, you can print the answers interleaved with the questions for your reference. The third option adds debugging information about how each question is arranged and the names of any graphics files used.

The Project menu allows you to create a new quiz, edit an existing quiz, copy or delete a quiz, combine two quizzes into one, or return to the operating system. Creating a new quiz and editing an existing quiz are functionally equivalent, so I'll only look at the creation option.

When you ask to create a new quiz, the question edit screen is displayed. The very top of the screen shows which question number you are editing and how many are currently in the quiz. Below this is a section containing the current question and answer. The question can be up to 255 characters in length, the answer up to 100 characters. The Giver program handles formatting of your question, but you still have limited control: the question may be placed at the top, middle, or bottom of the screen, and you can specify where new lines must start (for multiple choice questions).

Up to fifteen correct answers may be specified for each question. If you want to play Beethoven's most famous notes and ask "Give the number of the symphony from which these four notes are taken", you can tell Quiz Master to accept "Fifth", "5th", or "5" as correct answers. This, in conjunction with ignoring spaces and case, makes for a fairly flexible system.

Speaking of playing notes, the next block on the screen is associated with non-textual information. The Play String gadget is used to enter the music you wish to play. These are simple "computer tones"; no sampled instruments, but quite sufficient for a tutorial package. Quiz Master can play up to four note harmony over eight octaves, with time values from dotted whole to 32nd notes over a fairly wide range of tempos. The language used to specify the tones is fairly straightforward and easy to pick up; as an example, L1C+E+G indicates a C Major chord made up of whole notes.

The IFF Filename gadget is used for specifying the pathname of the IFF picture file to be displayed for the question, and here is where much of Quiz Master's power lies. You can create or digitize a picture of _anything_ and present it along with a quiz question. The picture can be created with any of the paint programs or digitizers; the only restriction is that all the pictures in a quiz must be of the same resolution.

Both sound and graphics may be presented with a question or after it. If the With option is selected, the sound and/or graphics are used to illustrate the question. If you want to display a graphic to show the student the correct answer, you would select the After option. Interestingly, with musically oriented quizzes Giver will play the student's answers. If the question is "Please build a E Major chord" and the pupil enters E+G+B, Giver will play the resulting E Minor chord for the student; if all the attempts are exhausted, Giver will play the correct chord and display the answer text.

Quiz Master allows you to clear the current question or delete it to an internal clipboard. You may also insert the contents of the clipboard at any point, which lets you rearrange question order. Also handy is the Retain gadget, which carries the current question over to the next slot. This would be useful for an ear training program: instead of typing "Is this a major or minor chord?" over and over, you need only enter the play string and appropriate answer for each question.

Once you've finished building all of the questions for your quiz (and you may have up to fifty questions per quiz and thirty quizzes per level), selecting the save gadget will bring up a requester and save your new quiz to disk. You are free to arrange questions and quizzes in any manner desired: you can use the eight categories as difficulty levels (as in Music Student), or you can put eight different subjects on one Giver disk.

When you've built your quiz, you can run Giver to make sure everything is as you intended. If you built your quiz on the Quiz Master disk instead of a student disk, you can make a copy of the Quiz Master disk and delete the Maker program from the copy, resulting in a student disk. After this step you can run the cleanup program to purge all the quiz files from your Quiz Master disk, enabling you to start a new project.

Wow, Deja Vu

Since Music Student was created with Quiz Master, and since Maker and Giver were written by the same programmer, it makes sense that they would have the same bugs. So, it was no surprise to find that Quiz Master doesn't like Fast RAM, or that many gadgets don't appear if you aren't logged into the Quiz Master disk. This could be more frustrating here than with Music Student, since the instructor is more likely to have an expanded system and be multitasking or swapping disks. Once again, you will have to toggle your Fast RAM off, and CD to the Quiz Master disk if working at the CLI level, before starting the program.

When you've finished creating a quiz, you click the save gadget to write the information to disk. But since the save gadget is simply a picture of a disk, my feeble mind didn't make the connection the first time, and I clicked EXIT instead. This took me back to the main Maker screen — and threw away all of my work. EXIT is separated from the other gadgets, but even so it's always easy to click the wrong gadget by mistake. It would have been a nice touch to add an "Are you sure?" requester; as it is, I'm willing to bet that at least one teacher is going to lose an entire quiz someday.

Quiz Master has an interesting option which I couldn't find in the manual: if you don't supply an answer to a question, it is treated as a tutorial and a Continue gadget is supplied in place of the answer box. This is demonstrated in the "Tutorial of Shapes of States" example: the student is shown a graphic of Maine, with the caption "This is the state of Maine" and a Continue gadget. This is repeated for Texas, then a graphic of Maine is shown as a question. The same procedure is then repeated for Maryland and Oklahoma.

This is an excellent approach to a tutorial, but it has one fatal flaw: the scoring mechanism does not handle it properly. It treats each tutorial as a question which was answered incorrectly. For the example above, the scoring box shows six questions (there were only two). Since a maximum of two of the "six questions" can be answered correctly, even a perfect student is admonished with a 33% success rate and the message "Better review the material!". Obviously, tutorial screens should be detected and excluded from the final scoring mechanism. Imagine spending a half hour interacting with the computer, working with a complete tutorial on the 50 states and answering all the questions flawlessly, only to be told that you don't know the material!

In Closing

Music Student, by itself, is a usable package — but that's about all. Because of the lack of associated documentation, it will either create a lot of unnecessary work for the concerned teacher, or much frustration for the students of a lazy one. If an instructor takes the time to properly integrate it into his or her course and remains aware of its quirks, it will most certainly be found useful.

In conjunction with Quiz Master, Music Student suddenly takes on a whole new character. A simple operation generates the list of questions which should have been included in the first place. More importantly, Quiz Master turns Music Student into a _starting point_. An instructor is no longer locked into the wording, order, or content of the questions, and is free to change anything as desired.

Of the two packages, Quiz Master is much more valuable, because with the Maker program an industrious teacher could write his or her own Music Student — and Geography Student, and Physics Student, and anything else needed for a particular class. Again, if the user is aware of the quirks they won't get in the way, although the problem with "tutorial mode" is saddening: the tutorial structure is very nice.

To ACS's credit, I will mention that I called them to clarify some information and, during the discussion, started listing the bugs I had found and things I didn't like. In many cases I only got halfway through describing a problem before they said "Oh, you mean...", proceeded to explain the bug to me, and indicated that the next revision would fix the problem. They seem concerned with giving their customers the best service possible, and some of the described changes support this impression.

That's going to do it for this month. Nybbles, Rick

•AC•

Associated Computer Services
1306 East Sunshine
Springfield, MO 65804

417-887-7373 (Information)
800-538-1032 (Orders)

Roomers

by the Bandito

Ed Note: You may notice that Roomers is a little less than chock full of juiciness this month. Well, summertime is always the Bandito's busiest season- - - On the road, hitting the various shows, scanning the Amiga globe. Don't worry though. Come Autumn, the Bandito will be back in full force and the rumors will be falling as quickly as the leaves!

Word has it that the toy operating system called MINIX is being ported to the Amiga in the Netherlands. This operating system resembles Unix, and is the working example of an operating system outlined in a popular computer science textbook.

Commodore may finally release a version of the Logo computer language because they hope to introduce the Amiga 500 into educational markets now dominated by the Apple II. The version of Logo would be Apple Logo compatible. Logo is a computer language in popular use for teaching children the concepts of programming.

Another rumor confirmed! Commodore engineers were never quite sure it would work, but they have created a new version of the Fat Agnes graphics chip used in the Amiga 500 and 2000. The new chip allows one megabyte of CHIP graphics memory. No word yet on how this upgrade graphics chip would be distributed. Chances are, it will be distributed this fall. Rumors still fly about higher resolution chips and expanded color capabilities. Some sources say "They will be out sooner than you think."

Graphics hack master Leo Schwab demonstrated 3-D glasses and stereo graphics demo programs at a recent FAUG meeting. (As usual, Schwab wore his flowing cape and hat. Some say that Schwab's employer is giving him free time to win the BADGE Killer Demo Contest, as long as the company name is displayed in the demo.) Several other companies are working on stereo glasses for the Amiga. The Forms in Flight animation program has a blue-red stereo pair mode that lets you view the animated objects in three dimensions.

An insider reports that the upcoming program from Master Designer Software based on the old Three Stooges movies has an opening similar to their Defender of the Crown game. Just like the old movies, the Three Stooges walk out in front of the "movie screen" and start the game. It has "nyuck-nyuck" digitized sound effects, too. No word yet on the actual plot of the game.

In England, the ComputerVision CAD company has ported their Medusa program to the Amiga. The product is in secret internal beta testing, and its release date is still under consideration.

Sun Microsystems has ported NeWS to the Amiga. It has been ported to several other microcomputers as well, but may be delayed due to possible lawsuits from Apple. NeWS is a windowing system that uses PostScript. You would see very dense graphics on your Amiga screen and then be able to print the screen at the maximum resolution of your laser printer. The Amiga screen is about 75 dots per inch, while most PostScript printers are about 300 dots per inch. They plan to show it to a select few at the upcoming SIGGRAPH show in Anaheim.

Dr. T, a well-known computer music software company, is porting several of their products to the Amiga.

The Amiga made the August 10 issue of Newsweek, under a caption "Desktop cinema: How to make a videotape at home." It mentions the use of the Amiga on the Max Headroom series, now being filmed for the fall season. It also claims you can use your Amiga to make home videos.

More details have arrived on the laser toaster project. Developers have been able to print HAM pictures with the "Jelly Jet." This mode is called "HAM on toast." They have also come up with an ANIM-compatible technique called "loaf animation" that renders the frames of an animation on sequential slices of bread. By flipping the toast slices in sequence, the animation can be seen. So far, these can only take place in brown-and-white, because the animation cannot be done in conjunction with Jelly Jet renderings because the toast won't fall properly once sprayed with colored jelly.

•AC•

The statements made in the Roomers column in no way represent the opinions or viewpoints of the authors or PiM Publications.

Oops... Corrections

The following corrections apply to the Amazing Computing™ Amiga Games Catalog listed in Volume 2.8. A complete re-listing of the companies who were victims of our errors has been included for your convenience.

MICROILLUSIONS

Land of Legends Available Soon

You're skulking around musty dungeons that just reek of danger. Keep your eyes peeled for secret passageways, stairs to other dungeon levels and teleporting devices. Be ready for rude meetings with frighteningly unfriendly beasts. If you need a break from your murky exploration, head into town and check out the shops and taverns. You can also design your own dungeon with your own traps, twists and turns. Soon to come is a Dungeon Construction Set which will allow you to dream up your own monsters, mazes, magic and more.

Faery Tale Adventure $49.95

The demons of the underworld are wreaking havoc throughout the land of Holm and have stolen the sacred Talisman of your home village. Only three young brothers have a chance at saving the countryside. Scour literally acres of land in search of answers to the gruesome reality that confronts you. Battle dragons, ogres, skeletons, and much more in your quest to destroy the malign Necromancer, the root of all evil. All done in spectacular scrolling graphics.

One-On-One Series - Turbo, Fire Power & Galactic Invasion Available Soon Three classic challenges - auto racing, tank battle and alien attack - are the first entries in this fast-paced, arcade style series. All three feature digitized sound/music and realistic graphics at a reasonable price.

CORRECTED ADDRESS

Microillusions
17408 Chatsworth St.
Granada Hills, CA 91344
800-522-2041 or in CA 818-360-3715

MEGATRONICS

Talking Trivia $19.95

Are you the local trivia king? Well, here are 2000 teasers to challenge your knowledge old and new. Add your own questions if you want to stump your friends.

Megatronics Inc.
55 N. Main
Logan, UT 84321
800-232-6342
CORRECTED IN-STATE NUMBER *801-752-2642*

Leather Goddesses of Phobos

A very funny, interactive fiction game from Infocom

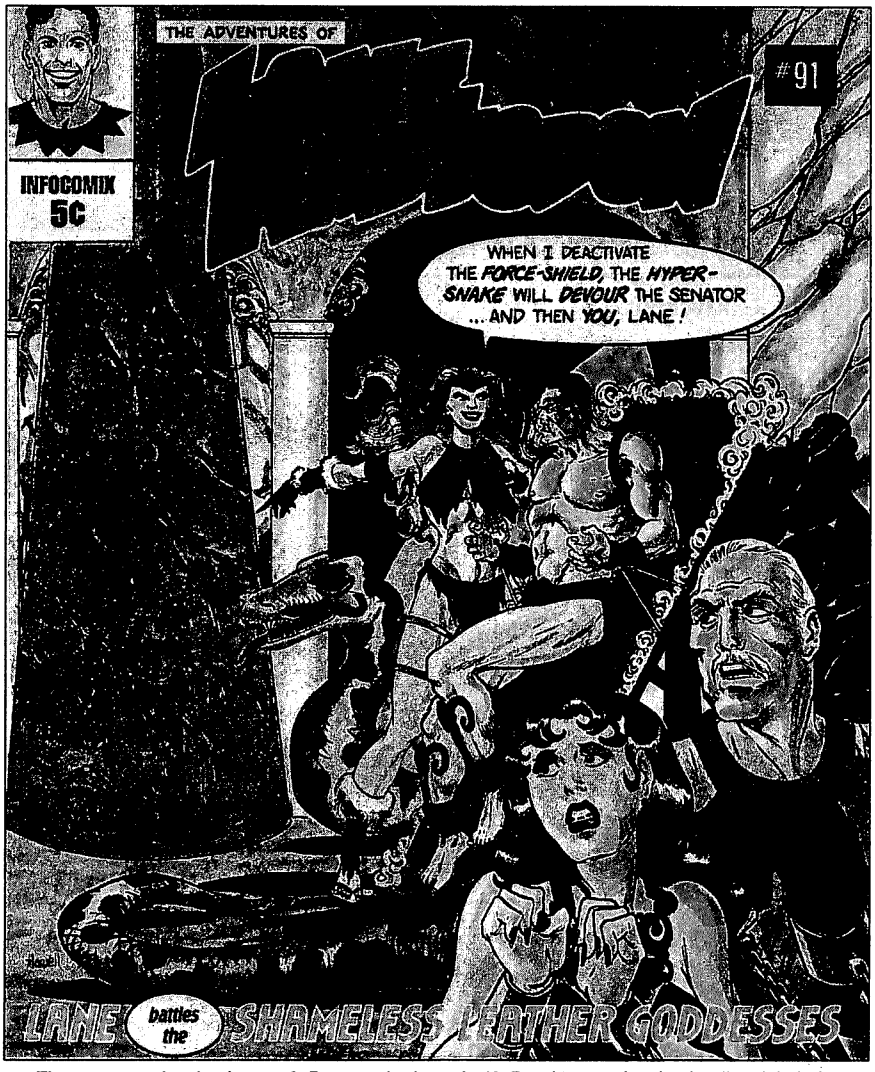
by Harriet Maybeck Tolly
Interplanetary Empress

Leather Goddesses of Phobos from Infocom is a text-based, interactive fiction game. For the uninitiated, this type of game provides a text description of your surroundings, what you can currently see, events that take place around you, etc. You, in turn, interact with the game by entering commands such as 'get on boat' or 'pick up jar.' Text fiction games have been around since the late 70's. This is not a game which shows off your Amiga's abilities; neither graphics nor sound are provided. This limitation, however, takes nothing away from the enjoyment of this very funny adventure.

At first glance, you might expect a sexist story line. Although it is true that your enemies are 'Leather Goddesses,' you determine your gender. According to your choice of restrooms at the start of your adventure, your character and your sidekick can be either male or female. You'll figure out what sex you've chosen by the clothes you're wearing. Throughout the story, other characters you meet up with will almost always be of the opposite sex from you. Women have an encounter with the Sultaness' husband, while men spend an hour with the Sultan's wife.

Even items you must acquire change according to your gender. Women must find a photo of Douglas Fairbanks; men need a photo of Jean Harlow. This approach is very refreshing in a sea of games which only know how to have a man save the earth from destruction. Leather Goddesses goes a long way to prove that a good adventure is a good adventure. The story line doesn't need to be modified for different players, just the point of view.

As the title suggests, an element of sexiness exists in this game. This area is handled so well, though, that no one should end up offended. Three modes are offered— tame, suggestive and lewd. Tame keeps the activity around hand-holding



The game includes a 3-D comic book (3-D glasses included) which is suggested reading before you begin your adventure.

level. If you try to engage your character in any sort of naughty behavior, you'll get a response such as 'Instead, you decide to get to know Sultaness' husband #1234 better, so you engage him in a stimulating discussion about the intelligence of beavers.' So, if you're really shy, stay in this mode and enjoy the adventure.

continued...

The Amiga Event!

Is Coming!

October 10–12, 1987
New York Sheraton Centre, New York City

SCHEDULE OF SEMINARS

Saturday, October 10th

	11:00 – 12:15	1:00 – 2:15	3:00 – 4:15
Conference A	Introduction to the A500	DeskTop Video Entry Level	Word Processing
Conference B	Introduction to the A2000	Graphics Professional – I	Music/MIDI Entry – I
Conference C	DeskTop Productivity Professional	A1000 Expansion – I	Entertainment – I
Conference D	Telecommunications	C and Assembly Programming	CAD Applications – I

Sunday, October 11th

	11:00 – 12:15	1:00 – 2:15	3:00 – 4:15
Conference A	DeskTop Publishing Professional	Avant Garde Amiga Art	Educational Software
Conference B	Music/MIDI Professional – I	Small Business Applications	Introduction to the A500
Conference C	Local Area Networks	DeskTop Productivity Entry	Amiga Graphics Entry

Monday, October 12th

	11:00 – 12:15	1:00 – 2:15	3:00 – 4:15
Conference A	DeskTop Video Professional – II	Introduction to the A2000	CAD Applications – II
Conference B	Music/MIDI Entry – II	CD–I and Optical Media	Engineering Applications
Conference C	Business Presentations	DeskTop Publishing Entry	A1000 Expansion – II
Conference D	Amiga Graphics Professional – II	Entertainment – II	Modula–2 and Forth Programming

Come to

The Amiga Event

AmiEXPO is a complete Commodore Amiga specific conference and exhibition, October 10–12, 1987 at the New York Sheraton Centre: three days of Amigan insight and information. Our Keynote Sessions, by leaders in the Amiga community, will highlight every thing from the origin of the first Amiga, to current software and the future of Commodore. The **AmiEXPO** Exhibition Hall is the heart of the show, featuring publishers, developers, and retailers from the entire spectrum of Amiga. Join us, for the only Amiga Event on the East coast, and the first of three National Amiga Events: New York – Los Angeles – Chicago. Become part of Amiga, become a part of **AmiEXPO**.

Exhibition Hall

A sampling of exhibitors:

Activision, Inc.
Amigo Business Computers
ASDG, Inc.
Brown–Wagh Publishing
Computer Living
Creative Microsystems, Inc.
Firebird Licensees, Inc.
Impulse, Inc.
Liquid Light, Inc.
MCP Associates, Inc.
Microillusions
NewTek, Inc.
Octree Software
PiM Publications
subLOGIC Corporation
Vertex Associates, Inc.
Amazing Computing
Ameristar Technologies
AmiProject
Associated Computer Services
Byte by Byte
Computer Systems Associates
Finally Software
Gold Disk Software, Inc.
Lattice, Inc.
Manx Software Systems
Meridian Software, Inc.
MicroSearch, Inc.
Micro Magic
New Horizons Software
PC Computer Solutions
Word Perfect Corporation

Keynote Sessions

Jay Miner, the Father of the Amiga, will open the New York **AmiEXPO**. **R. J. Mical**, the Designer of Intuition, will provide insights into software development.

For information call **800–32–AMIGA**
(in New York call 212–867–4663).

AmiEXPO Headquarters
211 East 43rd Street, Suite 301
New York, New York 10017

AmiEXPO Preregistration Coupon

☐ Yes, register me for AmiEXPO – New York!

Name _____
Company _____
Address _____
City _____ State _____ Zip _____

☐ One Day $10 ☐ Two Day $15 ☐ Three Day $20

Pre–Registration must be received by September 20, 1987.
After September 20 or onsite, there is an additional $5 charge.
Make check or money order (U.S. funds only) payable to: AmiEXPO.

AmiEXPO Associates
211 East 43rd Street, Suite 301
New York, New York 10017

'Lewd' is really a misnomer. This mode lets you end up actually making love with various characters in the story, but the descriptions are very far from lewd. They are simple and tasteful. "Oh," moans Sultaness' husband #1234, "say my number again... say it in French..." If you're worried about your kids getting their hands on the game, don't be. There is nothing here they havn't seen on prime-time TV. When I asked around for comments on the game, the answer was a resounding 'Not lewd enough.' The game understands most four letter words, so you can feel free to communicate with the characters in lewd mode. The response, however, always maintains the highest level of tastefulness. By the way, be sure to try using un-tame language while in tame mode. The response is quite witty.

For any Jerry Falwell followers, the references to him and the Moral Majority will probably be more offensive than the story. You're humorously warned of up-coming off-color language and instructed to consult the manual for the proper way to stop playing if you think you'll be offended.

The adventure is well laid out. As usual, you'll want to maintain a map of the places you've been. The map I created was remarkably close to the one provided in the hint package (more on that later). Since I'm not known for a strong sense of direction, I take this coincidence to mean the story provides consistent descriptions of locations. The puzzles facing you are, for the most part, solvable with some creative deductive reasoning. You'll need to stumble upon a few puzzles by accident, but everything manages to come together in the end. When you're stumped, the wit will keep you laughing. On your way, you'll encounter King Mitre. All he touches turns into angles.

The interactive mechanism is very good. The vocabulary is large and the parser is forgiving. For example, 'get off horse,' 'get down horse,' 'dismount,' 'dismount horse' and 'leave horse' all produce the desired effects.

Although Leather Goddesses of Phobos is a text adventure, it is far from one-dimensional. The game was awarded Best Packaging at the 1987 Excellence in Software Awards. The game includes 3-D glasses for a 3-D comic book (suggested reading before you begin your adventure). Also included is a scratch and sniff card. Various descriptions indicate a smell in the air. If you respond with 'smell', you'll be instructed to smell one of the scents on the card. After scratching number 5, I suggest you keep the card in another room from the one in which you are playing!

A map of one of the locations in which you could end up is provided. Without it you'd be eternally lost. Finally, beside the logic of map drawing, deductive reasoning and situation solving, part of your adventure will include encoded messages and word puzzles. For an all-text adventure, there is a lot going on.

If you have played other games where a fixed-total score is the only way to win, be forewarned. You'll soon find that points acquired for accomplishing different activities and total possible score are far from straightforward. Not to worry. If you manage to save the earth from the Leather Goddesses, point scoring irons itself out. As you accomplish certain tasks, your 'rank' changes. Rank ranges from Sandusky Stablehand to Princess of Pike's Peak, all the way up to Interplanetary Empress.

I pronounced myself Interplanetary Empress. Before you ask, I'll answer——No, I didn't solve the adventure without help. I did consult the hint package that Infocom sells separately. For those of you who think this is not a 'manly' thing to do, you're missing out on some of the best humor of the game. The hint book, a series of questions and answers, is hilarious. Wait until you are stuck in the story somewhere, then search the questions for some help. Not to be outdone by the multi-media approach of the game itself, the hints require a special pen to reveal the answers. Most questions provide a number of answers; the first being the most subtle, the last spoon feeding it to you.

There is truly good humor in both the questions and answers. One possible question you can ask is 'The scoring seems pretty cocka-mamie. How does it work?'. (Since Infocom considers it a hint, I won't tell).

Another question may be 'I'm not really picking up the 3-D in the comic. Can you give me some tips?'. The answer tells you to let your eyes adjust, then slowly move the comic towards and away from you. Finally, the answer states 'Where else but here could you develop invisible ink to learn how to read 3-D? What a wild and crazy bunch we are, eh?'.

You can ask 'HELP!!! How can I get out of Cleveland?'. The subtle answer — 'Millions ask this question daily.' Finally, if you ask 'What should I do in the Harem?', you're asked 'Do you really have to ask me?'.

A detailed map of all locations you may travel through is provided with the hint package. A cartoon character leads you around the map. She starts out by informing you that she's been to every hot spot from Venus to Vesuvius!

Leather Goddesses of Phobos has been a very successful product for Infocom. Additional awards include 'Best Adult Home Entertainment' from Peter Passel of the New York Times and 'Best Adventure-1986' from Britain's Newsfield Publications. These awards are well-deserved. This adventure provides hours (Who am I kidding? Months.) of laughs and puzzle solving, whatever mode you choose.

Special thanks to Mike Krowiak, a seasoned Infocom game player, for helping me through the universe.

About the author

Harriet Maybeck Tolly owns TollySoft, an Amiga software company in Wilmington, Massachusetts. (No, I don't sit around playing games all day. And, no, I'm not into leather.)

She can be reached at:
PeopleLink - TollySoft
BIX - rtolly
USENET - rtolly@CCA.CCA.COM
GEnie - TollySoft

•AC•

Leather Goddesses of Phobos, $39.95
(Non-copy-protected disk)
Hint package, $7.95
Infocom, Cambridge, MA

The Lattice C Compiler Version 3.10

reviewed by Gary Sarff

The Lattice C compiler from Lattice Inc. was the first C compiler available for the Amiga. This compiler still enjoys popularity on the machine, despite the advent of several other compilers from other vendors. Lattice C has been revised several times to fix bugs and add new facilities for program development. The current version is Lattice 3.10.

The Lattice C developer's package comes with several other programs. Most of these others have already been reviewed, so I'll only mention the names. The developer's package comes with the MetaScope debugger, the Lattice Make utility, the Lattice Screen Editor (LSE) and Lattice Text Utilities.

Lattice C comes on two floppies, one of which is bootable under AmigaDOS 1.2 and contains the executable programs that make up Lattice C. This disk includes two passes of the compiler('lc1' and 'lc2'), the assembler ('asm'), the linker ('blink'), the object module dumper ('omd') and the object module librarian ('oml'). 402 free blocks are left on this disk.

The assembler is for the programmer who wishes to code some time or space-critical routines, directly in 68000 assembly language. He can then link these routines, so they may be called from a C program. Unlike Aztec C, Lattice C still does not directly produce assembly language. Lattice C also does not allow assembly language to be specified inline in a C program.

Providing an assembler is, in itself, an improvement over past releases. Previous versions of the compiler did not include an assembler. It would also be nice if Lattice would allow inline assembler code and produce assembly language. Such moves would make some programming problems much more 'solvable, especially the transporting of C programs from other computers that do have inline assembly language in the program.

The linker is the new 'blink' linker from the Software Distillery. The new linker is a great improvement over the previous Amiga linker, Metacomco's Alink. Blink is noticeably faster and more efficient in its use of memory —— which is always in short supply with Amigas.

You can see the compiler's assembly language by using the object module dump utility. To run 'Omd,' take the name of an object file produced by pass 2 of the C compiler (the lc2 program). Such files traditionally end in the extension ".O." In the simplest case, type:

```
omd prog.o
```

After a few seconds, a symbolic disassembly of your compiled program is displayed. You can also make the 'omd' interleave your actual C source code with the disassembled output, to enable the location of some specific point in your program. Compile your program using the "-d" flag on the command line. This command instructs the Lattice compiler to insert "debugging" or line number information into the object file.

Omd can then read this information. You must supply the name of the source code file, along with the object module name to produce this interleaved listing. This feature can be very useful, particularly in tracking down programming bugs with the MetaScope debugger. "omd"'s interleaving can help non-68000-assembler-wizards narrow the area when tracking down bugs, to within a statement or two of the bug's location. Unfortunately, the output of 'omd' cannot be used as input to the Lattice assembler because format is not the same.

The object module librarian is also a new utility. The 'oml' is similar to utilities found under the Unix operating system for creating and maintaining your own libraries of precompiled routines that may be useful in your programs. The linker can search these libraries (just as it searches the Lattice-supplied libraries) for routines used by your program and then link these routines to form the final executable file. Such a utility has been sorely missed in the past. The user was forced to rely on public domain utilities.

The second disk, placed in DF1:, contains example source, hundreds of standard AmigaDOS header files and the C libraries the compiler needs to link with your compiled code to form an executable program. 52 free blocks are available on this disk. It is obvious that, for any substantial development, you must have a source disk for just your source code. This requirement begins the dreaded disk-swapping dance, since the Lattice system gives "assigns" to both disks on startup.

By pruning the provided disks, you can construct a C compiler system that just barely squeaks onto one disk. Guidelines for such work are listed in the Lattice manual.

You can work with the system without a hard disk. In the first pages of the manual, though, even Lattice recommends using a hard disk.

I was forced to change the assignment in the startup sequence for QUAD: to the RAM: disk. This switch tells the first pass of the C compiler to output its intermediate code (known as "quads") to a temporary file on the Amiga's RAM: disk. If you are compiling large programs or have other programs running at the same time, you may not have enough RAM to hold the quad file. The Amiga has been known to crash when the RAM: disk runs out of room.

Although this space problem was much worse under AmigaDOS 1.1, it has not been eliminated entirely. This difficulty is hardly Lattice's fault, though; it's AmigaDOS's fault. I set the QUAD: name through the AmigaDOS ASSIGN command to a disk file. Lattice does not recommend this. . . and with good reason.

You should avoid setting QUAD: to the same drive as the C source file because the compiler constantly reads your source program and then writes to the quad file. The second pass of the compiler constantly reads the quad file and writes out object code. Both the compiler's passes produce much disk head movement and seek through the AmigaDOS file system, which is known to be slow. If at all possible, you should assign the quad file to either RAM or a floppy other than the source code disk. Even the extra floppy can cut compile times in half!

continued...

The manual

Another crucial item in a compiler package is the manual. If the manual is not workable, the compiler will not be workable either.

Some incorrect examples at the beginning of the manual may confuse the inexperienced C and AmigaDOS user. Pages 4 through 8 of the manual discuss the various Amiga directories and path assignments that the startup disk makes (or the user needs to make) to use the system. Many of these elements are wrong, or at least incompatible, with the startup-sequence file supplied on Lattice disks. For example, when the manual is supposed to be taking the user through compilation of example programs, it instructs you to enter:

cd LC:source

AmigaDOS responds, "Can't find LC:source," which is sure to confuse the first time user. A quick check of the startup-sequence file shows that LC: was assigned to C:, so the example, as printed, won't work because the example source files are on DF1: in a subdirectory called "source." Sure, this problem is minor, but what about new users?

Otherwise, the 400+ page manual is well done. The wire-looped binding lets the pages lay flat on your desktop. The manual is nicely indexed, giving the name of each library routine, its purpose, the page number and a classifying type.

The type may be "Unix," meaning this routine is a library routine with the same name as a Unix C library routine that does the same thing. Type may also be: "Amiga" (an Amiga specific library routine), "Lattice" (a Lattice C specific routine probably also available on their other C compilers such as for MS-DOS), "XENIX" (this routine is from Microsoft's XENIX operating systems C compiler) and "ANSI" (indicates the routine is part of the proposed ANSI C standard). This indication of types is very helpful for the programmer who wants to write portable code. . . it also adds a nice touch to the manual.

The library routines are nicely described in strict alphabetical order. The name of the routine, a "Synopsis" section (usually consisting of a list of any needed header files or argument type information), a description of what the routine does, what values (if any) the routine returns and a "See also" section (that may list the names of other related routines) are all listed.

The manual also discusses portability——a good thing for programmers to know, even if they do not intend to move their programs to other machines. A working knowledge of this stuff simply helps the programmer avoid poor or non-portable, non-standard practices.

Library routines

Aside from assignment statements, flow control statements and variable declarations, C programs are mostly calls to library routines. The strength and number of library routines provided with any C compiler is a good measure of the "power" (what can easily be done with the compiler).

Lattice C provides about 259 different library routines (not including innumerable routines that are part of AmigaDOS, which are also accessible from C) which can be called from C. Many new routines of interest have been added since the most recent release of Lattice.

Lattice C has added a new math library, allowing easy access to the AmigaDOS fast floating point (FFP) routines. These additions are the same routines Aztec C uses——the ones which gave Aztec C its reputation for producing fast programs.

Also added was support for IEEE format floating point routines provided in an AmigaDOS library. Two different methods may not be used in the same program, since there are actually two separate math libraries and only one can be linked with a program.

There are new "level 0 I/O functions" which support low level Amiga I/O, using the AmigaDOS "file handles." These functions can bypass the higher level 1 and 2 I/O functions and deal directly with AmigaDOS. The programmer must do some extra "bookkeeping" type work, but his programs will be a bit more efficient in terms of I/O, since there will be less library overhead.

Some new functions have been added to allow access to AmigaDOS file information, including file attributes and the date/time stamp that AmigaDOS attaches to files when they are created or modified. Functions are also available to access the current directory path.

Many functions have been added simulating Unix functions, including functions dealing with time: asctime(), ctime(), gmtime(), localtime() and tzset(). These functions let you obtain ASCII strings containing date and time information, as well as month names, days of the week and even allowance for daylight savings time and different time zones.

Other Unix-compatible additions are mkdir() and rmdir() (for creating and removing AmigaDOS disk directories), qsort(), an implementation of the "quick-sort" algorithm for sorting arrays of any type of data), signal() (which allows Unix-like "signal" or exception processing) and an enhancement of the printf() function (to allow for new format specifications and removal of the 200-character limit in the output string, found in previous versions of Lattice). I am glad Lattice provided these routines; I hope they will provide more such routines in future releases.

The way aggregate objects (such as structures, arrays and unions) are passed to called routines has been changed since the last release of the compiler. Previously, when you attempted to pass an aggregate type by value:

Example 1

```
struct node {
  int count;
  char type;
  int value;
};

int p1(s)
struct node s;
{
...some code
}

main()
{
  struct node thisnode;
...
  p1(thisnode);
}
```

The compiler pushes the address of the aggregate onto the stack, "thisnode." In the routine p1(), the compiler generates code to take this address and copy some number of bytes (equal to the size of the aggregate) to some temporary storage inside p1(). This process simulates the process of "calling by value" used by other C compilers. The Unix C compiler (and others) do things differently. At the point that p1() is called, other compilers actually generate code to push the actual bytes of "thisnode" onto the stack. The only noticeable difference between these two methods is if you are interfacing assembler to C, or otherwise attempting to access the 68000's stack or registers directly.

I did come up against this problem while working on a project. Everything halted because the ported program was from Unix and it was not worth rewriting thousands of lines of code which depended upon thie Unix method. I am happy to say that Lattice C now follows the conventions of the other C compilers and pushes the actual byte values of the aggregate onto the stack before calling a routine.

Differences from Unix C

While there are several "standards" for C, (the Kernighan and Ritchie Standard, the ANSI C standard, etc.) there is also what may be called the Unix C standard. Tens of thousands of C programs have been developed to run under Unix. These programs often can be compiled and run without modification on computers as diverse as Sun workstations, PDP-11'S, IBM 370's and Cray 2 supercomputers.

This flashiness is not as impressive as it seems. What we are really talking about is only one C compiler, the Unix Portable C compiler, which runs on all these computer systems. Once the compiler has been ported to different computers, the programs written for that compiler will probably run on these computers. The Unix C standard is then in the same class as other standards, such as the IBM-PC and MS-DOS. Unix C is a standard by fiat, a standard because it is numerous. Still, we must deal with a few processing "differences" between Lattice C and Unix C.

These differences are not operating specific (such as the lack of certain functions or system service calls that are available under Unix). There will always be differences between any two systems of such diverse capabilities and, in any case, these functions and calls are not part of the C language itself.

The differences discussed here are things that are important to know if you plan to try to move C programs from another system to the Amiga. Such movement was common activity in the early days of Amiga, since there was so little software available for the machine. This work remains popular for C work on the system.

Example 2
Handling complex typedefs.

```
1. #include <stdio.h>
2. typedef struct atom {
3.    char p1[20];
4.         long p2;
5. } atom;
6. typedef atom atomtable[];
7. atomtable t_atom = {
8. "ATOM 1",4L,
9. "ATOM 2",8L,
10. "ATOM 3",16L
11. };
```

Lattice C does not handle typedefs of arrays or structures in the same way as the Unix portable C compiler. Example 2 defines a structure called atom. In an attempt to reference this structure as a new type, Example 2 defines a new type called atom, instead of having to say 'struct atom'. Line 6 then attempts to define another new type, atomtable, as an array of atoms. At this point, Lattice C runs into problems. Line 6 generates error number 67, illegal object. Line 11 generates cryptic error number 35, closing brace expected. This error confused me because a closing brace is already present. The solution to this problem is rather simple. Just rewrite line 6 to read:

continued...

```
6. typedef atom atomtable;
7. atomtable t_atom() = { ...
```

Experienced C programmers realize that line 6 is redundant and can be simply removed. A further problem pops up, though. Wheile attempting to transport a C program, all these problems must be sought out with a text editor and rewritten, so the program can be compiled.

Example 3

```
#include <stdio.h>
int globe=0;
int incr()
{
  globe++;
  return globe;
}
main()
{
  printf("%d %d %d\n",
    incr(),incr(),incr());
}
```

Example 3 shows the difference, in order of evaluation of arguments to a function between Lattice C and Unix C. I have tested this program on many Unix systems, including VAXen, Suns, PDP-11's and others. Also, the answer printed by this program is:

3 2 1

This response proves that arguments are pushed on the system stack from right to left. Manx C also prints '321.' After compiling with Lattice C on the Amiga and running this program you see:

1 2 3

This response seems to indicate that Lattice pushes arguments from left to right——a direct conflict with the Lattice manual. The manual explicitly states that the compiler pushes arguments from right to left.

Examination of the machine code produced by the compiler reveals what's happening. Lattice proceeds down the argument list from left to right and determines if an argument is "complex" (is it a function call or expression). If the argument is "complex," it is evaluated and the result is saved in a temporary location. If it is a simple constant (variable or literal), that argument is skipped. When the end of the argument list is reached, Lattice returns down the list from right to left, pushing either the simple arguments or the values of any previously computed complex arguments. You can see this strangeness by producing a function call with both constants and function calls as arguments. I can think of no reason for this bizarre way of doing things.

It is true that the venerable C bible, K&R, states that the order of function evaluation is up to the compiler implementor. The Unix standard is from right to left and Unix is certainly not an unknown operating system. A great deal of Unix code exists that actually depends on this evaluation order. This practice is admittedly very poor programming, but when you try to port a complex program, it is nice to avoid rewriting the program as you go. It is interesting to note that so many microcomputer C compilers claim to be Unix compatible (whatever this new buzzword means) and how many fail tests like this, which have nothing to do with simulating Unix system calls but are simply plain, albeit poorly coded, C.

Optimized code

When Aztec C became available for the Amiga, Lattice gained a reputation for producing bloated and slow machine language. In fact, Lattice does accomplish some optimization of the machine code produced by the compiler better than other widely used compilers. For example, I wrote a small segment of C code declaring a two-dimensional array of structures and some integer variables and pointers. I then constructed a very complex expression using these integer variables and pointers as subscripts to the array.

Upon examination of the code produced by the Portable C compiler, I was shocked to find that the compiler generated code to evaluate this complex expression every time it appeared as a subscript. The Lattice compiler, a compiler costing considerably less money, generated code to evaluate the expression once and stored the result in a temporary location. When the expression appeared again, Lattice simply loaded the value from the temporary location.

Even previous versions of Lattice C performed such optimizations. Although I have not seen the newest release of Aztec C, Lattice makes more efficient use of the 68000's register set. (Ed note: The Aztec C compiler is reviewed elsewhere in this issue.)

Support

The first page of the Lattice manual mentions the different kinds of support available to registered owners. You may obtain technical assistance through the BIX network, a McGraw-Hill sponsored network associated with Byte magazine. On BIX, you can trade electronic mail directly with Lattice personnel.

Lattice also maintains its own electronic bulletin board service, which may be reached 24 hours a day at (312) 858-8087. This bulletin board allows users to find out about Lattice patches, to to work around known bugs. The bulletin board is also your outlet for informing Lattice of bugs that you may have discovered.

Lattice also has a technical support hotline where owners may talk with Lattice support personnel about bugs or usage problems. Lattice warns that this line may be busy at times and you may be forced to leave information on an answering machine and wait to be called back.

The vast numbers of Lattice C users on the many national computer networks and the hundreds of electronic bulletin boards are also a valuable support resource. Often a user may post a question or problem and have a response, often several responses, within a few hours or a day.

Conclusion

Every C compiler has its strengths and weaknesses, but, on the whole, this version of Lattice C has many strengths. The few "weaknesses" can be worked around. Depending upon your needs and budget, you should consider the Lattice C compiler.

•AC•

Manx 3.4a C Update

by John Foust

As recently as a year ago, program development tools for the Amiga were nonexistent or, at best, very primitive. The Lattice compiler was slow and the Metacomco linker was slower still. Very few people had hard disks or memory expansions.

Developing a large program is very time consuming on Amiga floppies. In fact, many Amiga programs were cross-developed on MS-DOS machines, using a version of the Lattice compiler that runs on MS-DOS computers, but creates program files that run on the Amiga. The developer transfers the compiled program to the Amiga via the serial or parallel port.

A few months after the Amiga arrived, it was followed by the Aztec C compiler from Manx Software. In compile times, Manx C was, and still is, much faster than Lattice C. The Manx executable program file was much smaller than programs made with Lattice C. The Manx environment has many more freebies (such as utility programs) than the Lattice environment. Manx C is also more Unix-like, thus attracting many programmers.

Today, the situation has changed. Amiga developers are no longer limited by their hardware. The soft "chuff-chuff" of hard disks and and the silent speed of a large recoverable RAM disk have replaced the unending "gronk-gronk" of floppy-disk-based C compiles. Lattice cut the fat from its library routines and improved its coding techniques and is now much closer to Manx.

Manx has added features to its latest release of the compiler (version 3.4a) but a series of bugs have tainted its release. An update is planned to correct these bugs. According to Jim Goodnow, the wizard behind the Manx compiler, version 3.4b compiler was sent out for duplication in late June (Ed note: Version 3.4b was shipped during the third week of July.). The MS-DOS Amiga cross-compiler will be available at that time as well.

The Manx Aztec C compiler comes in three flavors. The lowest level is called the Professional system which includes the compiler, the assembler, a linker and the Amiga include files. The Developer system includes all these features in addition to a series of Unix-like tools such as 'make' and 'grep', object module librarian utilities, an editor and a debugger. The Commercial system also has the above features, plus the source to all library functions and a year of free updates.

The 3.4a commercial update came with a half-inch of additional manual pages for the Manx binder. This documentation is a summary of the improvements in the software (The opening boilerplate still claims Fred Fish will reproduce his disks for free. I'm sure Fish does not appreciate this kind of publicity.).

The 3.4a compiler has several new features, including a new code generator, to help the compiler produce faster, smaller code than before. Enumerated types are supported. The Manx 3.4 compiler does not yet support the emerging ANSI standard. A new Lattice compatibility mode is also included. The Manx compiler defaults to 32 bit integers, and otherwise works like the Lattice compiler. This compatibility makes code conversion much easier.

The floating point library support now handles three formats - Motorola FFP, IEEE double precision emulation and IEEE double precision hardware emulation, if a 68881 is present. The assembler now supports opcodes for the 68010 and 68020 processors, as well as in-line code for the 68881 floating point coprocessor. The assembler now understands Metacomco assembler directives. Previous to this development, the Manx assembler and the Metacomco assembler were incompatible.

Manx has also tinkered with object module format and code methods. Previous incarnations of the Manx compiler could not make certain types of programs (such as printer drivers) because of code differences between the Manx, Lattice C and the Metacomco assembler. Now, Manx 3.4a can produce printer drivers.

The Manx linker now supports scatter loading in memory. Previous versions produced one large hunk of code, instead of small pieces. The linker supports Amiga (or Metacomco) format object modules as well and detects the object module type automatically.

The Manx libraries have been improved. The functions that can invoke other programs, the 'fexec()' family, have been notoriously unmanageable in previous Manx versions. Because of tricky programming, the 'fexec()'-type functions stopped working between AmigaDOS 1.1 and 1.2. Fortunately, not many programs used these functions, so few programs were incompatible. The new versions work well under 1.1 and 1.2, and should continue to work in the future.

In general, Manx obviously is working on improving the Unix compatibility of their libraries. 'ioctl()' is now present and the 'stdio' functions have been improved. You can choose from several ways of buffering to the console, making standard I/O more flexible.

New debugger

A new debugger, shipped with advanced versions of Manx, has multiple windows, one for each task you are debugging. The debugger has a fairly powerful macro ability-- - to create macro functions that dump structure contents in an orderly way, for instance. A new function trace feature displays the functions that are being executed, with the listing indented to indicate the depth of nesting. This feature is switch-selectable from the compiler. Using the debugger adds a little overhead to generated code, but the new features are quite worthwhile.

One of the neatest new commands in the debugger is 'ap'. 'ap' recovers from a Guru-- - sort of. If a Guru appears and 'db' can recover, it does and reminds you that the program is still alive. You then enter 'ap' and it searches the list of tasks and asks for the task number to debug, as well as suggesting a task number that might be corrupt. Click CANCEL on the Software Error system requester and the program comes right back - maybe, or maybe not alive.

In some cases, recovering the machine after a Guru is impossible. There is no way to free the memory and system resources used by the dead task. In AmigaDOS, resources are not tracked, so open files

continued...

remain open and memory is not deallocated. The 'ap' feature gives you a few moments to back up files in the RAM disk, if you haven't been saving your work.

The new 'al' command waits for a task to be loaded and takes control of the new task. I discovered a problem using the Metacomco Shell with the debugger. When I used the 'al' command to intercept a program loaded in another CLI window, it would never find the new task. According to Goodnow, this fault cannot be attributed to the debugger. The debugger detects a new task in the system by intercepting the LoadSeg() operating system call. The Metacomco Shell is undoubtedly redirecting the vector as well, or doing the LoadSeg() for a new task within its own code, thus preventing the debugger from capturing the code.

Among the new utility programs included with Manx is 'ctags', a program that filters a C source file and returns the names of function declarations. In conjunction with a command in the Manx 'z' editor, you can jump to a given function's source code, no matter which file it is in. 'Z' is Manx's clone of 'vi', a favorite Unix editor.

The Manx 'include' files are approximately five to ten percent smaller than the Lattice 'include' files, even though they contain the same information. The Lattice files contain more white space than the Manx files. This type of economy is a nice touch which saves a fair amount of space on work disks.

The source code to the Manx libraries is included with the commercial version of the compiler. The code is archived in a Manx-specific format and strangely enough, the archive program does not compress the source text. Rather, it just concatenates the files together with a little glue. I imagine this method was used to save space on the disk because many small files take up extra blocks on the disk, as compared to a single large file of the same information. In the process of de-archiving the library source, I discovered all the source fits on a single Amiga disk, with room to spare. This point made me wonder why the files were archived at all, especially considering the time it takes to de-archive all the files.

3.4a bugs

But what about the bugs in 3.4a? Soon after 3.4a was released, Goodnow posted a patch program to Usenet and several other networks. Three patch programs were posted in succession, each subsuming the previous patches and adding more.

The patch file solution has its faults. Sure, the patch fixes problems in the compiler and allows less buggy development to continue, but how does an owner find out about the most recent patches? It should be safe to assume that a newly-minted compiler, fresh from the company is in working order. . . but this was not the case with 3.4a.

Both Lattice and Manx use the patch program method to adjust their compilers after shipment and before the next release. Over the course of this review, I talked to many Amiga developers about other topics. I asked if they had seen patches for the latest Manx compiler. Unfortunately, several had not, and had complained about bugs in 3.4a. I sent them the patch program and hopefully helped solve their problems. It is disheartening to consider the implications a few bugs killing the development of the next wave of Amiga products.

Not every developer spends time visiting the computer networks or bulletin boards where these patch programs are posted. It can be argued that all serious Amiga developers should visit at least one network, especially a network such as BIX, where official Commodore technical support resides. Reality proves otherwise. Many developers are busy at work on new versions of their program and can't spare the time. The patch program solution and implementation leaves me unsatisfied. I think both Manx and Lattice should consider sending a postcard to notify owners of the bug fixes. Anything less means wasted time and effort on the part of the compiler owners.

When I solicited comments on the national computer networks concerning the Manx 3.4a compiler, many nervous people questioned its reliability in code generation, even after the patch programs were posted. I could not verify all the claims, so take this impression with a grain of salt.

According to Goodnow, "The serious bugs were fixed with the patch program." These bugs included problems with using a constant array index larger than 32768, and declaring arrays with more than 32768 elements. The compiler's parser stored array size limits in short integers, leading to the conflict. Goodnow suggests a work-around by allocating the array with standard memory functions.

Goodnow did indicate a few problems in the linker. There were other bugs in 68881 code generation and in over-optimizing bit-field expressions. Another bug generated code that treated byte array indexes as signed integers, preventing access to elements greater than 32768.

Test suite

A compiler is a complex piece of software. A useful method of testing a compiler is a 'test suite.' A test suite is a set of programs that test the features of the language. Test suite diagnostics can pinpoint bugs faster than field-testing. After 3.4a, Manx purchased a C compiler test suite. Goodnow spent several weeks passing the suite through the 3.4a compiler. The test programs revealed all the reported bugs and discovered several others. All these bugs are fixed in 3.4b, according to Goodnow.

There are two additions to Manx 3.4b. The first addition affects computers with 68020s, such as the CSA Turbo Amiga. While 3.4a added 68020 support in the assembler, the compiler code generator did not produce 68020 code. Manx 3.4b has 68020 support in the compiler-generated assembly language generated as well. The second added feature is the ability to directly compare structures.

Support

Manx Software maintains a conference on the BIX network called "manx.aztec" that gives online support to Manx users on all computers, including the Amiga. Manx also has a bulletin board system at the New Jersey offices. Both Goodnow and Manx technical support people are online. Judging by the messages on the board, both give prompt and complete answers.

According to Manx update policy, Manx will send the update disk to owners of the commercial version. Developer version owners will receive a letter of notification, informing them that they can get the update info for a nominal fee.

Manx is on the brink of releasing version 4.1 of its MS-DOS compiler. This version includes a number of improvements, including ANSI C compatibility, a source level debugger and a new front-end. Because the same main code is used for all Manx compilers, the features in this version will soon be ported to the Amiga. Goodnow hopes to have the Amiga 4.1 version of the compiler available by the fall of 1987. Goodnow also notes that the Amiga source level debugger should be available before the fall, however.

A compiler release with bugs is unnerving at best, and a serious waste of program development time at worst. According to my own informal survey, while many developers were very unhappy with the bugs in the Manx 3.4a release, very few abandoned it for the Lattice compiler. Many developers are still impressed with the Manx compiler's speed and ease of development.

•AC•

Jim Goodnow II

Developer of Manx Aztec 'C' for the Amiga.

by Harriet Maybeck Tolly

AC: How did Manx get started?

Jim: "Basically, I have two partners: Tom Fenwick and Harry Suckow. Thomas and I both started working at the same company at the same time. He had written a C compiler for the 8080 for the Heath.

At the same time that he [Fenwick] was working full-time for this company, he was also working part-time with Harry. They had a partnership and did consulting work on the side. When Tom finished the compiler for his Heath machine, Harry said, "You really ought to sell that" Tom said "OK" and they turned it into a product. They started to bootstrap themselves with an initial ad in 'S-100 Micro Systems' for a small amount of money and started selling compilers. When Tom made the compiler run native (it was originally a cross-compiler from the PDP-11) and tried to actually put it on the 8080, it didn't quite fit. So, he wrote a sort of p-code interpreter and a p-code compiler. He compiled the compiler with the pcode compiler, so it would all fit.

It was at this point that I got involved. Once he had an interpreter, it was real easy to write a 6502 version of the interpreter and produce a compiler that would produce code for an Apple II. This is the machine that I had and was playing with at home.

So, over a period of time, sales slowly ramped up. Not really fast, because we were all working full-time and weren't spending a lot of money on advertising, but sales did continue to go up. It reached the point where Harry, who was working full-time, got behind enough in shipping that he decided to take a week and catch up. Previous to this, we had an answering service taking a lot of the calls and keeping track of customers. Harry would then pick up the numbers and on the weekend, put together a bunch of boxes and ship them out. He fell behind enough that he decided to take a week off and spend it putting things together and sending them out. Since he was there that week, he was also able to answer the phone. Since he was able to answer the phone, sales doubled! So, he had all these extra compilers to send out. He stayed around another week and sales doubled again. So, he sort of never went back to work.

I think a lot of this had to do with the fact that the answering service was run by elderly ladies that didn't quite understand the concept of "C compilers" as they called them, or computers in general. It wasn't the best environment for sales. As time went on, sales increased. Eventually, we got mentioned in a *Byte* article, which helped our sales a lot. So, we started advertising in *Byte*, which helped more.

I went full time in 1983. A few months later, Tom went full time. We worked at home on MS-DOS machines (S-100 machines with 8086's in them) and did cross-development. A year later, we hired Chris Macey to do a new code generator for the 8086; a highly optimized code generator.

At that time, we bought a PDP-11 and got a real office. We use the PDP-11 now for most of the real work. We've got a real office for the three of us, with nice paneling and carpeting and stuff. Meanwhile, the rest of the operations moved from here to there. We had a history of moving. We were in an old funeral parlor, in the back of a computer store, sharing a desk with the guy who owned the store. About six or seven months after that, we took most of one building and part of another building. We closed the one office that the development people had and combined the whole company together. A little over a year ago, we moved into a space that was even more condensed."

AC: How big is Manx now?

Jim: "We have about 30 people working at Manx now. Our development staff has grown quite a bit. We have three people dedicated just to technical support. There are currently about 50,000 users of our C compilers out there and annual sales are about 4-5 million. Although we've grown a lot over the past three years, we still try to provide the support you would expect from a smaller company."

AC: How would you describe your current position?

Jim: "Pretty much, I am the driving force behind the 68000 product line of Manx Aztec. I did the original 6502 code generator. When the Mac came out, I was very excited. I was convinced it was going to sell. We decided to make an investment - take a risk - and produce a compiler for that machine as well. I ended up spending 4 months, 7 days a week, 20 hours a day doing the Mac, because when it came out, there were no development tools for it other than Lisa Pascal and cross-developing. So, it seemed to me that being out there first was real important. We put a lot of time into it.

At the end of that time, I felt I needed something significant out of the company. I'm not totally money oriented - I decided I wanted to go to California. It was something tangible that was good for me. When we started the company we started with different intentions. But the idea was for each of us to get out of it what we wanted. What I wanted out of it was freedom, so I was able to move to California nine months later. I've been doing a lot of my work out here, in cooperation with people on the East coast.

At this stage, I primarily do Amiga-specific stuff, but also 68000 generic stuff. When I do a new code generator, (such as for the new front end parser that they'll do for the PC) and move it to the 68000 on the Amiga, I give it to other people who move it over to the Mac and to the Atari. So, it just happens that since I'm doing it on the Amiga, it ends up there first, but the work I'm doing is actually across all of the 68000 product lines. The compiler breaks down into a front end which parses the language and a back end which generates code. I've had little to do with the parser; that's all handled by Tom. When they finish a new parser for the PC, we move it over to the 68000. The parser puts out a tree that's used by the code generator. I'll take the parser that they've done, which has all the latest things like ANSI standard stuff, and attach the 68000 code generator."

AC: Describe your current development environment.

Jim: "The original work was done as a cross from MS-DOS to AmigaDOS, mostly because I had a machine with a 10MHz 8086 and a 40 Meg DMA hard drive. As soon as I had a reasonable hard disk, I

continued...

switched to native development. When we shipped V3.2, the librarian was able to read, but could not create libraries very well. Although libraries were created on the cross system and downloaded, we never actually used the librarian native. So, using the stuff native is much more robust. You are much more likely to find problems and eliminate little glitches. When you actually use the stuff, you notice when something is annoying. So, you fix it. When you're doing it cross, a week before you ship it, you decide to test native. You don't always notice things that are missing or aren't the way they should be.

Currently, I'm running with a CSA Turbo-Amiga (68020 and 68881) which really speeds things up. I have 2 Mbytes of MicroBotics StarBoard2 RAM and their MAS-Drive20 Hard Disk. I do want one of the DMA hard disks, hopefully taking advantage of some of the file system changes that speed things up. I also want to be able to put the hard disk into the CSA box. I've got this expansion box that has space for it. Instead of having three boxes, I'll just have two. What I really want is an A2000 to put it all in there! I'm on a standard Amiga, using the CLI. I haven't played much with the shell programs, but I'll probably switch to one of those soon because of some things that I can't do in the CLI."

AC: What other benefits are there in doing development on a 68000 for a 68000?

Jim: "Clearly, there are differences between the 8086 and the 68000, in terms of word and byte ordering. For example, what we used to do when we did cross development was when we'd ship, we'd put the stuff on the native machine; [we'd then] compile it native, so that we knew we could bootstrap and make sure it worked correctly. It is interesting that before we shipped V3.4, I uploaded everything to my MS-DOS machine, cross compiled it, and found a few bugs that didn't show up native! So, having both actually works out well. The other reason I went native to the Amiga was that it was the only machine, besides our PDP-1, that is multi-tasking. When I first read the Exec documentation, I was reminded of an operating system I had worked on with Tom that was just like it. I knew this stuff was state of the art. It was not the Commodore 64; this was real stuff."

AC: What are some negative aspects to working in a native environment?

Jim: "The biggest problem is that I only use one machine, so I'm in danger of clobbering my hard disk while testing programs. Without some sort of protection, such as having a second machine to test on, there is a potential for problems. So far, I have been OK."

AC: What is the biggest change for people who upgrade to version 3.4 of the Aztec 'C' compiler?

Jim: "Version 3.2 was put out too early. It went out incomplete. It didn't support scatter loading, double precision floating point or overlays. My intents were to basically fill up these holes and provide a nice, solid, complete package that had all the pieces that were missing from V3.2 and to provide them to everyone who had V3.2. The biggest changes were in the linker. It now supports 4 types of scatter loading, so programs don't have to be one big contiguous hunk. It also supports overlays, via segmentation, so that programs that are very large, such as DPaint II, can be run on a machine with only 512K bytes of memory, by loading in segments when they are needed. (DPaint II and MaxiPlan are both done with Aztec C; there are actually products out that were compiled under beta versions of Aztec!).

The other area was the floating point. We supported Motorola fast floating point by default. We didn't support the double precision. This was a miscue on my part - I didn't think they had a double precision library on the Amiga. There was a library on V1.1 that was called 'mathieeedoubbas.library'. When I saw the 'bas', I thought, oh, this is some special interface for Basic. I didn't read the docs well enough, so I didn't even bother to try to implement double precision. We didn't have an emulation at that point and I didn't think the Amiga had theirs yet. So in V3.4, we have implemented the fast floating point, IEEE floating point and 68881 hardware floating point inline code. In the process of doing that, we made some major changes to the code generator, primarily because when you deal with the IEEE emulation, you have to talk about register pairs. You have to talk about D0, D1 as a pair. The register allocator I had before really didn't do that [deal with registered pairs] very well.

So, at the same time, we improved the code for fast floating point. Some programs, like Aegis Draw, went from 190K to 150K because of the changes I made to fast floating point. Before, it just put out a pseudo instruction and the assembler expanded that. This involved saving some registers, doing some things, then restoring them. Now, the compiler can be very smart about what registers to save or not [save]. There was a lot of garbage that got generated. This is now eliminated. The floating point stuff involved major changes to the code generator to support the three different formats.

Once the code generator supported the 68881, we had to put support in the assembler for the 68881. Since we were doing the 68881, I decided that since the 68020 is coming on strong, we should put support in the assembler for the 68020, as well as the 68881. Adding the 68020 opcodes, plus the 68881 opcodes added a lot to the tables. The 68020 adds additional addressing modes. The table started getting really big. So, we had to redesign the tables as well. So, the whole effort, which was supposed to be two weeks, took two months, which is the biggest factor in the delay. We should have cut it off without the 68881 and 68020 support.

The assembler now supports all the 68881 and 68020 stuff. At the same time, I put in support for all the MetaComco directives, so that people could use the assembly language header files that come from

Commodore directly and also type in the examples from the ROM Kernel manual. All sorts of things that are done in assembly language with the MetaComco assembler should be do-able with the Manx assembler. It's about 95% complete."

AC: Do you see C staying as the Amiga developers' language of choice, or do you think other languages such as Modula-2 might become more popular?

Jim: "First, there aren't a lot of options in other language compilers; even for C, there are only two. I think that as the Amiga becomes more popular, that will change. As more companies move their compilers from other machines to the Amiga, you'll have more choices. This will make it possible for people to come up with a Pascal or Modula-2 as a reasonable alternative. Right now, I think that of the compilers out there, C seems to be generating the best code. It seems to have come from more competitive environments. The IBM PC market is incredibly competitive, with MicroSoft pouring money into it. The companies involved are constantly pushing their technologies to the limit. To the extent that a company is involved in both the PC and 68000 market, that push of technology will spill over; whereas, if you're just starting out with a compiler for the Amiga, you're probably not going to be at the same level. As more companies see the Amiga as a viable market, and move their stuff over, you'll see more high-powered code generators.

In terms of what people will be using, there is a strong bias in the industry towards C, especially in the PC market. It's been more portable than Pascal and portability is a big issue. A lot of the public domain programs on the Fish disks came off of Usenet and were Unix programs. They weren't written for the Amiga at all. There is a lot of power in that. The only significant changes I see are in the direction of the ANSI standard and possibly C++. This [change] gives you the power of C, but also the data abstraction and some of the more exotic features of other languages."

AC: Do you see Manx getting into any other language compilers?

Jim: "We have talked about doing a front end for FORTRAN or Pascal. We have no plans; we just talk about it [expansion to other languages] as being something that, when we've run out of things to do..."

AC: What other directions does the company have, other than compilers?

Jim: "We have done some publishing of third- party software. We sell some third party library and windowing type packages. We are in a position, since we deal with a lot of people who use the compiler, that if they like our compiler and trust us, we can publish their packages."

AC: What kind of work has Manx done with the ANSI standard?

Jim: "Tom Fenwick, who does the parser for all the compilers, has been on the ANSI standards committee for several years. We are about to release our new compiler for the PC. Thi [compiler] will support the ANSI standard changes. As soon as we finish it for the PC, we'll move it over to theAmiga. Those changes are especially important for the Amiga because of the 16-bit and 32-bit interface problem. With the ANSI standard idea of prototypes, we can basically eliminate the whole confusion. People can run with 16-bit integers by having the prototype set up correctly."

AC: Any final words of wisdom for Amiga 'C' developers?

Jim: "I'm excited that Commodore seems committed to making the Amiga real. The new machines show that Commodore's future lies in the Amiga and in bringing all those C-64 people up through the A500 to the A2000. It's still more bang for your buck than any other machine out there."

•AC•

Moving?

Subscription Problems?

Please do not forget to let us know.

If you are having a problem with your subscription or you are planning to move, please write to:

Amazing Computing™
Subscriptions Questions
PiM Publications Inc.
P.O. Box 869
Fall River, MA 02722

Please remember, we can not get your magazine to you if we do not know where you are.

Allow 4 to 6 weeks for processing

AC-BASIC

For programmers who want to create stand-alone applications that offer good performance, and who like the ease of use they get with BASIC

by Sheldon Leemon

Everyone knows that 'rea'l programmers don't use BASIC. Interpreted BASIC programs run slowly, are difficult to copy-protect, and are hard to use as "stand-alone" applications, since they need the BASIC interpreter program in order to run. For these reasons, most commercial programs on the Amiga are written in C, Modula-2, or assembly language. BASIC programs just don't get much respect in the marketplace.

And yet despite it's deficiencies, BASIC remains one of the most popular, if not the most popular, programming language. For one thing, it is significantly easier to learn than C, particularly for non-programmers. You don't have to devote weeks or months to learning the ins and outs of the compiler and the Amiga ROM Kernel routines in order to program in BASIC. It's possible to sit down and write your first program in a matter of minutes.

The Amiga Microsoft BASIC dialect is particularly powerful. It gives the novice programmer access to powerful user interface features such as pull-down menus and mouse pointer control, without requiring him to delve deeply into the mysteries of Intuition. And the process of writing a BASIC program is much simpler than using a compiled language like C.

Because user programs are run within the framework of the BASIC interpreter program itself, it is possible to type in a program using the built-in editor, and get immediate feedback on whether the program works or not. The interactive nature of interpreted BASIC makes it very easy to modify the program and see the results, which is very important, since most programs require dozens of modifications before they're finished.

For programmers who want to create stand-alone applications that offer good performance, and who like the ease of use they get with BASIC, AC/BASIC offers the best of both worlds. It's a BASIC compiler which takes standard Amiga BASIC programs, and turns them into machine language programs that can run from a Workbench icon or from the CLI, without loading the BASIC interpreter. Because these programs don't require the overhead added by the interpreter, they can run many times faster than ordinary interpreted BASIC programs.

The compiler process

As with any program, the first step to producing an AC/BASIC program is to enter the source code. This is the actual program listing, which in the interpreter, appears in the "List" window. With the interpreter, you create a program by typing it in line by line in the List window. AC/BASIC programs, on the other hand, may be created with any kind of text editor, including the one that is part of the interpreter.

The interpreter usually saves a program in a compressed format, so that the code does not appear readable if you examine it with a text editor. AC/BASIC requires that programs be saved in a straight ASCII text format. The interpreter allows you save a program this way, by using the "A" option. To save a program called "Myprog" to an ASCII file, for example, you would type:

```
SAVE "Myprog",A
```

in the command window. Some word processors also allow you save a program as straight text, without special print formatting instructions. With Textcraft, for example, you would use the "Text Only" save option.

Once you've created your program, you may test it by running it under the BASIC interpreter. It won't run as fast as the compiled version, of course, but it will let you know if you've committed any syntax errors while entering the program. When the program is running satisfactorily, and you've saved it to a text file, you're ready to compile it.

To compile your program, you must load the AC/BASIC compiler. You may do this either from the CLI, or by clicking on the program icon. Once you've loaded the program, you must select the source file to compile, by choosing the "Open" item from the "Project" menu. At this point, a screen will appear, listing the various compiler options. These are used to control how the program will be compiled. The various options are:

Use Long Addressing. Most of the time, the compiler will use the relative addressing mode in order to make the program shorter. If the program must reference data or make a jump farther away than a 32K displacement, however, it will need to use absolute addressing instead. The compiler will tell you if this option is needed. It makes the program larger, and in rare cases, a bit slower. Absolute addressing may be used selectively by including a "$OPTION +A" command line in your program.

Enable Run-time Tests. This option causes the compiled program to perform more extensive internal error checks on the values of variables, and the intermediate results of math and string expressions. It will detect integer underflow and overflow, array subscript range errors, and stack errors, for example. Since it slows down program execution significantly, it can be used during development and removed in the final version of the program.

Compile for Decimal Math. While the interpreter only uses floating-point math, compiled programs can use Binary Coded Decimal math instead. This type of math is slower than floating point, but it allows for greater precision, and eliminates rounding errors.

Generate Errors List. This option can be used to send error message to the list file, rather than the screen.

List Include Statements. This option can be used to add statements in "included" files to the program listing.

Generate Full List. This option generates a printed list with symbol table and label cross reference.

Process Run-time Events. This option is used to include the code which performs the special processing needed to handle event-driven commands. These are the commands which take the form " event ON", such as MENU ON, MOUSE ON, BREAK

ON, TIMER ON, ON ERROR, etc, and all of the OBJECT statements. Since this code does a lot of processing during the vertical blanking interval, it slows down the program significantly. If time is critical, you can structure your program to do polling, instead of using event-driven functions.

Link Run Time. Normally, the compiler only produces an object file, which cannot be run without also loading library files. If this option is used, however, these library files are linked to the object file to produce a stand-alone program. Since linking the library files expands the program by about 40K, you'll probably want to produce only the object file during program development, and then link the libraries in on the final version.

Generate Symbol File. This option is to be used in connection with a symbolic debugger. Since this debugger is not yet available, the option is useless for now.

Temporaries on RamDisk. This feature tells the compiler to put its temporary files on the RAM disk to speed things up. Unless memory is limited, you'll always use this.

Default Arrays to STATIC. The interpreter always uses dynamic arrays, which can be changed in size and redefined. The compiler allows the use of static arrays which are faster, since they can only be declared once, and may not be erased or redefined. They can't be used for graphics, however, on a machine with expansion RAM, since they will be loaded into FAST RAM rather than CHIP RAM.

In addition to the options listed above, the compiler allows you to select the amount of RAM allocated to it for a Work Area. It starts with a minimal amount of RAM, which you can expand if necessary. If you wish, you may save your settings, which will, in subsequent sessions, come up as the default set. It should also be noted that any of the compiler options may be invoked from a command line, if you start the program from the CLI. Using this mode, the compilation takes place with no further intervention required from the user. There's even a batch mode which allows you to compile several programs with one command.

Once you've set the options, you start compiling by clicking on the "Compile" button, or pressing the Return key. The compiler goes through three different passes, and gives you a report as it goes on. Statistics, such as the number of lines processed, the amount of storage space used for labels and symbols, the size of the object file produced, and the size of the stack required to run the program are listed. You need to know the latter only if you plan to run the program from the CLI. If you run it from the Workbench, the stack size is set automatically, but from the CLI, the STACK command may be needed to adjust the stack size for a large, stack-intensive program.

Another statistic supplied is the compile time in minutes and seconds. The speed of the compiler is generally quite good. For example, from floppy disk, a 100-line program takes about 20 seconds to compile without linking in the run-time library, and about a minute and a half when the run-time library is linked. Since much of this time is fixed overhead, longer programs compile even larger numbers of lines per minute. A 2500 line program takes about four and a half minutes to compile on floppy disk, and about two minutes from the RAM disk. If you have enough RAM to put your source code file, the compiler, and compiler libraries in the RAM disk, compiler speed is very fast indeed.

Documentation

The manual that comes with the AC/BASIC compiler is quite thorough. It's about 300 pages long, and includes sections on getting started, using the compiler, the compiler options, batch compilation, and the differences between the compiler and the interpreter. There is also a complete BASIC language reference that covers all of the Amiga BASIC commands. Some short program examples are included on the program disk. At the back of the manual, there are appendices on internal data formats, ASCII codes, reserved words, machine language programs, statement differences between the Macintosh and Amiga versions, run time errors messages, and compile time error messages. There's also an advanced installation guide which is very helpful for customizing your installation, depending on your drive configuration. The manual includes a complete index.

While it's not exactly what I would categorize as light reading, the manual is quite thorough, and it's language is not so technical as to be beyond the understanding of the average user. It does neglect to mention one or two details, however, such as that the "RUN" command, which in the compiler is used to chain another program, must be executed when the default window is active, rather than a window on a custom screen. It would also have been nice if they added a check list of things to look for when compiling a program written for the interpreter.

How Compatible Is It?

One of the main reasons for buying this compiler is to make existing Amiga BASIC programs run faster, so the question of

continued...

compatibility is an important one. Absoft has done a very good job of seeing to it that compiled AC/BASIC programs run just like the interpreted version, only faster. This is not too surprising, since Absoft had previously written a BASIC compiler for the Macintosh in cooperation with Microsoft. The company is therefore quite familiar with Microsoft's 68000 BASIC language, and has a good working relationship with Microsoft, who wrote the Amiga BASIC interpreter.

Most of the differences between compiled BASIC programs and those running under the interpreter are due to the inherent differences in the method of execution. Interpreted programs are executed in the order in which the program flow is directed. If you write a program that starts like this:

```
GOTO Skip DIM a%(50), b(100) Skip: PRINT "We skipped the DIM statements"
```

the interpreter would not execute the DIM statement, and the arrays a% and b would not be dimensioned. The compiler, on the other hand, creates the program in the order in which it is written. Statements like DIM and DEFSNG only tell the compiler to set aside a specific amount of storage space, or how much space to assign to variables. These statements, therefore are treated like compiler directives. In the above example, the DIM statement would cause the compiler to set aside storage for the arrays, even though this statement technically never "executes". For this reason, it is safest to put all DIM and DEF statements at the beginning of the program. That way, the compiler and interpreter versions will act the same way.

Other differences are inevitable. For example, some immediate statements like LIST, LLIST, LOAD, SAVE, STOP and CONT are not applicable to compiled programs. The FRE and CLEAR statements are a bit different also, since the compiled program allocates as much memory as it needs to run, and thus doesn't differentiate between free program memory and heap. The compiler limits arrays to 7 dimensions. It allows STICK and STRIG functions to be used for either mouse port, not just the second one.

The OBJECT commands run slightly differently on compiled programs than they do on interpreted ones. For one thing, the interpreter misses object collisions from time to time, while that bug doesn't show up in the compiler. There are also slight differences in floating point range and accuracy. Programs that use "FOR...NEXT" delay loops to create pauses will not work the same under the compiler, since the delays generated will only be about a tenth as long.

Some of the differences allow compiled programs a bit more power than interpreted ones. For instance, SUB programs don't have to be declared STATIC, which means that they can be recursive. SUB programs must, however, be listed at the end of the program, and cannot be interspersed with normal program code as they can in the interpreted version. Absoft includes a SUBSORT program that sorts your source code to place sub programs at the end. Another addition to the compiler is a SELECT CASE command, which implements a command very similar to Pascal's CASE or C's switch statement. The compiler also allows what Absoft calls "metacommands". These are statements like IGNORE, INCLUDE, PAGE, and OPTION, which allow you to control the compilation process, including turning compiler options on and off from within the program.

Once you are familiar with the few small differences between the compiler and the interpreter, it isn't very hard to make the few changes necessary to compile a program written for the interpreter. What I generally do is to load the program text into an editor with search capabilities, and locate all DIM and DEF statements. I move these statements to the top of the program. Next, I make sure that SUB statements are at the end of the program. Finally, I check for the keywords that operate differently, such as RUN and CLEAR. I make whatever adjustments are necessary in statement containing these commands.

Using this method, I was able to compile a number of programs that were written for the interpreter. I found that I usually had to make only one or two changes, even to quite long programs. For example, to compile a 2500-line PAC-MAN program, I only had to change one RUN statement, and remove a call to the library routine WaitTOF (which conflicts with the compiler's vertical blank routines, and crashed the program). About half the programs that I tried compiled correctly without requiring any changes.

In the course of testing the compiler, I ran into one or two bugs. For example, you can't call ROM Kernel library routines from a SUB program, and the PRINT USING command doesn't always work right. Absoft is aware of a number of these bugs, and has included a sheet of suggested work-arounds with the compiler. They plan to come out with a maintenance update for the compiler soon, which will correct some of the problems. Considering the complex nature of the Amiga BASIC language, it's surprising that the first release of the compiler has as few bugs as it does.

How fast is it?

The bottom line in BASIC compilers is always "How much faster does it make the program run". Here too, AC/BASIC does very well. The sample programs that we compiled ran from 2 to 10 times faster than the interpreted versions. The variation in speed increase is due to the fact that BASIC programs include many different types of operations, each of which benefits from the compilation process in varying amounts.

For example, the program shown in Listing 1 draws 100 rows of 200 dots each. The interpreted version takes about 51 seconds to execute, while the compiled version takes only about 22 seconds to execute. If we remove the single PSET command, however, which

in both versions makes a call to the ROM Kernel WritePixel routine, the resulting empty loop program takes 10 seconds for the interpreted version to execute, and less than two seconds for the compiled version.

By changing a single line, the speed advantage for the compiled version grows from 2.5 times as fast to over 5 times as fast. That's because the compiler isn't much faster than the interpreter when it comes to making system calls like WritePixel, but is a lot better at reducing the overhead associated with program loops.

Any way that you want to measure it, we found the speed improvement afforded by the compiler to be quite noticeable. For example, a shareware Pac-Man game written in BASIC went from unplayably slow to unplayably fast when compiled. For those of you who still believe in benchmarks, I've thrown in a couple of the classics, the Sieve and the Byte calculations benchmark. As the results in Table 1 show, the compiled version of these programs run from 4-5 times as fast as the interpreted versions.

Summary

The AC/BASIC compiler is a solid piece of software which compliments the Amiga BASIC interpreter quite well. Although the $200 price tag is fairly high as BASICs go, it's not too bad when you compare it to the price of Modula-2 or C compilers. Absoft requires you to sign a license agreement before distributing programs which contain the AC/BASIC run-time code, but the company doesn't charge any fee for this license, which means that you can distribute your programs freely once you've signed and returned the agreement.

A product like AC/BASIC can only help increase the Amiga software base. A lot of specialized applications, such as genealogy programs, astrology program, bowling-league statistic programs, etc., are first written in BASIC and then compiled. Even some popular commercial software, such as the DAC-Easy accounting package is written in compiled BASIC.

The similarity between IBM Basic and Amiga BASIC is great enough as to make porting IBM BASIC programs a relatively simple task. The availability of AC/BASIC will make it very attractive for IBM BASIC programmers to port their products to the Amiga. And, of course, AC/BASIC will also allow even relatively inexperienced Amiga programmers to develop applications whose performance compares well with those developed in languages like C. Finally, the BASIC programmer may be able to get a little respect.

Table 1
Benchmark Timings

	Amiga BASIC	AC/BASIC
PSET Time (secs)	50.82	21.96
Empty Loop Time(secs)	10.06	1.92
Sieve Time (secs)	61.68	11.46
Byte Calculation Time (secs)	18.84	4.89
Byte Calculation Error	-5.96E-08	-1.79E-07

Listing 1: PSET.BAS

```
StartTime = TIMER
   FOR x = 50 TO 250
      FOR y = 50 TO 150
      PSET(x,y)
      NEXT y
   NEXT x
PRINT TIMER-StartTime "seconds"
INPUT "Press Return to End";x
```

Listing 2: SIEVE.BAS

```
10 LET StartTime = TIMER
20 LET Size = 7001
30 DIM Flags%(7002)
40 PRINT "Start One Iteration"
50 LET Count = 0
60 FOR i = 1 TO Size
70 LET Flags%(i) = 1
80 NEXT i
90 FOR i = 1 TO Size
100 IF Flags%(i) = 0 THEN GOTO 180
110 LET Prime = i + i +3
120 LET k = i +Prime
130 IF k >Size THEN GOTO 170
140 LET Flags%(k) = 0
150 LET k = k+Prime
160 GOTO 130
170 LET Count=Count+1
180 NEXT i
190 PRINT "Done: ";Count;" Primes Found"
200 LET FinishTime = TIMER
210 PRINT FinishTime-StartTime "seconds"
220 INPUT "Press Return to end";i
230 END
```

Listing 3: CALC.BAS

```
LET StartTime = TIMER
LET nr = 5000
LET a = 2.71828
LET b = 3.14159
LET c = 1
FOR i = 1 TO nr
    LET c = c*a
    LET c = c*b
    LET c = c/a
    LET c = c/b
NEXT i
PRINT "Done"
LET FinishTime = TIMER
PRINT "Error = ";c-1
PRINT FinishTime - StartTime;"seconds"
INPUT "Press Return to End";i
END
```

•AC•

AC-BASIC Compiler

Part I

reviewed by Bryan Catley

During the past three weeks, I have had the opportunity to experiment with Absoft Corporation's AC/Basic Compiler Release 2.1. In this issue, I intend to provide an overview of the product and a few general impressions. In the next issue, we'll get down to some more specific information.

The suggested retail price for AC/Basic is $195, but I have already seen it advertised for as low as $165. You receive a three-ring binder, which is slightly smaller than the three ring binder you received with your Amiga. Inside is a non-copy protected disk, a registration card and one of the most complete sets of documentation I have ever come across. Not only are the compiler and its language extensions FULLY documented, but a complete AmigaBasic reference section WITH LOTS OF GREAT EXAMPLES is also included. There are also a couple of extra pieces of paper. A licensing agreement which you must sign and return to Absoft if you intend to distribute software developed with the compiler is included (There are no royalties required. Absoft just wants to know that you are doing it). The other is a program patch which we'll discuss later.

What's on the Disk?

The AC/Basic disk contains a number of goodies. First and foremost is the compiler itself. The disk also includes a utility program named SortSubs, a sample program named "Hello" and a bunch of drawers containing all the examples shown in the text of the named chapters- - - A very neat feature!

Use AC/Basic from Workbench or CLI

AC/Basic may be executed from the Workbench environment simply by double clicking on its icon. It may be executed from CLI by typing name, optionally followed by the name of the program to be compiled.

What can be Compiled?

The obvious answer is: "An AmigaBasic program, of course!". Unfortunately, even though that response is essentially correct, it is not quite that simple. Yes, you compile an AmigaBasic program, but it MUST be in ASCII format. This restriction means that every program you compile must have been saved from the interpreter using the following command: SAVE "name",A — you may not use the Project Save and Save as items to accomplish the necessary save. (Forgetting to do this usually results in a compile time error on the first line of the program). Of course, you may also use any other editor to enter and save programs.

Further, AC/Basic also allows you to do batch compiles. Essentially, you may compile as many programs as you wish with one invocation of the compiler. To make use of this facility, you create a small ASCII file (use the AmigaBasic interpreter if you wish) which names the programs to be compiled and provides a list of the desired options (see below). Then, you must tell the compiler it is working with a batch compile and point it at the program list!

Compile Time Options

Any compiler worth its salt provides you with a complete set of compile time options, each designed to serve a specific purpose and to provide as much completeness and flexibility to the user as possible. AC/Basic is no slouch in this area, providing the following options. These options may be selected individually or in combination, when any program is compiled.

A -Use Long Addressing
Do not use this option unless the compiler instructs you to use it.

C -Enable Run-time Tests
Duplicates many of the execution time tests made by the interpreter.

D -Compile for Decimal Math
By default, the compiler uses binary for floating point mathematics duplicating the approach used by the interpreter.

E -Generate Errors List
Error messages are written to disk.

I -List Include statements
Include statements are listed in a full program.

L -Generate Full List
A full program list, including error messages (if any) is written to disk.

N -Process Run-time Events
This option MUST be selected if the program to be compiled contains any event processing statements such as "ON event GOSUB label", "event ON/OFF/STOP", etc.

R -Link Run-time Library.
The run-time library is combined with the program at compile time.

S -Generate Symbol Table.
Generates a program symbol table. This feature is primarily for a future (I believe) symbolic debugging system.

T -Temporaries to RamDisk
The compiler is to use RAM disk for its work files.

U -Default Arrays to STATIC.
AC/Basic provides a STATIC extension to AmigaBasic for handling arrays.

By default, options C, N, and T are in effect. They are activated because C and N most closely approximate the interpreter environment, while T speeds up the compile process considerably.

Compiling from the Workbench

Double click on the "ac-basic" icon to compile. The compiler loads very quickly and presents a Copyright Screen (please respect this copyright — if you don't, you'll hurt not only yourself, but every other Amiga user as well).

At this point, you have a single Project menu to work from with "Open," "Print" and "Quit" items. "Quit" terminates the compiler. "Print" allows you to print listings previously generated by the compiler. "Open" allows you to select a program to be compiled via a requester.

Selecting Compiler Options

Once a program name has been entered in the requester, you are presented with the compiler's Control Panel. The lower portion of the panel lists all the options described above. To select any option, just click in the small box to its left and the option becomes highlighted; click a second time to turn the option off.

At the bottom of the panel are two gadgets labeled "Save" and "Clear." "Save" saves all your selected options for use as the new default option set. "Clear" simply clears all options to "off."

The top of the panel contains the name of the program you wish to compile, along with four gadgets. Click "Compile" to compile the named program, click "Compile Batch" to initiate a batch compile (the given name must be of a program list, rather than an individual program) or "Cancel" to forget the whole thing. The fourth gadget is a "sort of" slider which tells the compiler how much memory it may use as a work area. (This work area size is saved along with all the other selected options when "Save" is chosen).

Compiling from CLI

The compiler is just as easy to use from CLI. Just enter a command line based on the following format:

AC-BASIC program-name
<compiler-options>

The compiler-options may be entered with or without spaces between them. Note the availabilty of two additional options when the compiler is invoked in this manner. "Wn" allows you to specify the amount of work space memory to be made available (at least one space must always follow "n"). "B" informs the compiler that a batch compile is being requested. As far as I can determine, there is no method for saving a new set of default options from CLI.

Compiler Metacommands

Metacommands are special compiler commands which are embedded within the program, which are not part of the program itself. Since they are not real basic statements, metacommands must be prefixed with a $ and embedded in REMark statements (This structure allows the program to function with the interpreter). The general format of a metacommand is:

```
REM $metacommand <arguments>
' $metacommand <arguments>
```

AC/Basic supports the following four metacommands:

REM $IGNORE <ON | OFF>
All statements following an $IGNORE ON are ignored by the compiler, until an $IGNORE OFF is encountered.

REM $INCLUDE "filename"</L>
The compiler reads statements from the ASCII file "filename" and treats them as if they are physically part of the program currently being compiled. If "/L" is specified, the included statements become part of the list file, regardless of whether "l" was selected as an option.

REM $OPTION <+ | -> <A | C...>
This metacommand allows you to selectively turn the "A" and "C" compile time options on and off during a compile. The option is set if a "+" or a blank precedes the option. The option is reset if a "-" precedes it.

REM $PAGE
A page break is inserted in the program listing.

AC/Basic Language Extensions

There are three distinct AmigaBasic language extensions in AC/Basic:

STATIC Arrays If any or all arrays are DIMensioned with an integer (rather than a variable), and are never ERASEd or reDIMensioned, they may be declared STATIC. STATIC arrays may be formed for individual arrays within a program (e.g. DIM STATIC array(20)) or for all arrays by requesting the "U" compile time option. The result is smaller and faster programs.

Dynamic Subprograms These subprograms are just like ordinary subprograms, except they lack the STATIC keyword in the SUB statement. Dynamic subprograms allow recursive calls (i.e. they may call themselves), while static ones may not allow such calls.

SELECT CASE Statement This statement may be thought of as an enhanced form of ON ... GOTO/GOSUB ... Implementation is a very powerful and a full description is beyond the scope of this article. However, the following simple example provides an idea how SELECT CASE might be used:

```
  INPUT"Enter a number:",value%
  SELECT CASE value%
    CASE 1 TO 5
msg$="Value 1, 2, 3, 4, or 5 selected"
    CASE 6,8,10 TO 20
msg$="Value 6, 8, or 10 to 20" selected"
    CASE ELSE
msg$="No checked for value selected"
  END SELECT
  PRINT msg$
```

Remember, if any of these extensions are used in a program, the program will no longer work with the AmigaBasic interpreter.

Compiler Interpreter Differences

As might be expected, there are a number of differences in the way certain statements are handled by the interpreter and by the compiler. There are not as many discrepancies as might be expected, though! Some of the more obvious ones are:

• subprograms must physically appear at the end of the program. For those programs which have subprograms scattered throughout, AC/Basic provides the AmigaBasic SortSubs utility to rearrange the program.

• DEFtypes (such as DEFDBL, DEFINT, etc) are processed as they are encountered. This arrangement means that statements such as "IF a=0 THEN DEFINT a-z" are essentially meaningless to the compiler.

• Function declarations (DEF FN) are also processed as they are encountered. The compiler expects to physically find the declarations before they are used. (I.E. Don't include them in a subroutine which physically appears at the end of the program).

• OPTION BASE is only allowed to appear once within a program. It must physically appear before the first DIM statement.

• Mixing numeric variable types (x, x%, x&, etc) in computations may produce differing results or an error. (Use of the "C" option during development catches these conditions).

• The following statements are not recognized by the compiler: CONT, DELETE, LIST, LLIST, LOAD, MERGE, NEW, SAVE, and TRON/TROFF (all quite reasonable, if you think about it).

• END, STOP and SYSTEM are all treated identically by the compiler (again, as you might expect).

• There are minor differences when using the CHAIN, RUN and COMMON statements.

• SOUND statements at the end of the program may not be heard for the duration (Include a delay loop to compensate).

• FRE(-1) and FRE(x) return the same value with the compiler.

• CLEAR performs its initializing functions, but any parameters are ignored (they become unnecessary).

• The STICK and STRIG functions work with either port, or both simultaneously.

continued...

- FOR ... NEXT delay loops produce a MUCH shorter delay in a compiled program. Use the TIMER statement to produce an appropriate delay. For example, the following code fragment causes a delay of 10 seconds, regardless of whether the interpreter or the compiler is used:

```
wait&=TIMER
WHILE TIMER<wait&+10:WEND
```

Patches

Many existing bugs may be fixed by providing REGISTERED users with patches. In fact, one patch came with the version I received (It fixed a problem with INSTR). The patch consists of the source listing of a patch application program and the patch itself. To use the patch: Enter the patch program and the patch as ASCII files. Then, run the patch program (it works as either an interpreted or compiled program).

Current Problems

There are two basic (no pun intended) problems I have encountered so far. First, some programs require even more memory than the "normal" basic versions when compiled. This problem means that some programs which just barely fit into memory with the interpreter will not work in a compiled version (incredible, but true). Second, if the work space allocated with the "Control Panel" is not large enough, an "Out of Memory" message appears. However, the work files are not closed, meaning you cannot do another compile, nor may you even delete the work files. You must re-boot before you can re-invoke the compiler.

There are other problems. One of the most annoying is the inability of compiled programs to recognize every mouse click. Sometimes, several clicks are required before a click is acted upon. Another irritation is the rather startling system crash at the conclusion of some programs (These crashes usually involve programs which make direct use of operating system functions)!

Please note that I have contacted Absoft regarding these (and other) problems and even sent them examples of every problem I have encountered. They have been very responsive to my questions and I have no reason to doubt that solutions to all these problems will appear before long. (Once again, if you purchase the compiler and don't register your purchase, you won't receive the fixes).

Should You Purchase the Compiler?

I would love to give you an unconditional yes, but right now, I cannot. If you consider yourself a serious AmigaBasic programmer, and you don't mind a few frustrations (or adventures!), you will gain many benefits from the compiler and it will probably be worth your investment. If you don't fit into this group, I suggest you wait for a release or two before buying the compiler. By then, most (if not all) of the more serious problems should be corrected. If you do make the purchase, you'll have what surely will end up as being one of the most important software products ever released for the Amiga.

Ultimately, I believe every AmigaBasic programmer will want a copy of this compiler. Why? The compiler will provide a combination of the convenience of interpretive development and the power of a final product which has been compiled. The drawback of this ideal arrangement? The only negative I can think of is the inability of Basic to support secondary menus! Still, with a good basic compiler, a good basic programmer can to give C programmers a real run for their money!

In preparation for the day when you do buy Absoft's AC/Basic compiler, there are a couple of things you should do to avoid some of the initial adventures (or frustrations):

DIMension ALL arrays. According to the manual, this work should not be necessary, but, without exception, I have found it to be necessary.

Use the $IGNORE ON/OFF metacommands around ALL the CLEAR statements in your programs. Once again, this work should not be necessary, but you'll be better off!

Next Time...

We'll discuss the output from the compiler, provide some compilation times (very impressive) and look into some of the current problems in a little more detail. In the meantime, a couple of requests to Absoft: How about some icons for the ".lst" files, so Workbench users can easily clean up after themselves? How about including the Patch program on the distribution disk (as SortSubs is), so users do not have to type it in?

Finally, if you feel adventurous, go ahead and buy the compiler and mail in the registration! Apart from a few frustrations, you won't regret it.

•AC•

Directory Listings Under AmigaDOS

Why They're So Slow, and How to Make Them Faster

by Dave Haynie

bix: hazy

usenet: {caip|allegra|ihnp4}!cbmvax!daveh

The Amiga System can often be an interesting mix of amazement and frustration for the typical user or programmer. The amazement usually comes around when you discover that the operating system already provides a facility for the strange and wonderful thing you're trying to do. The frustration come when you try to do something you think of as a simple, typical task, and the job appears more difficult, more involved, or just not as efficient as you were expecting.

The Amiga CLI is different than the command-line interfaces on other computers. A CLI user has to use the #?()| characters in his wild card specification, the * character isn't a wild card, and you can't use ^S and ^Q for display start and stop. So obviously there must be something wrong with CLI. The programmer is required to ask to use a system resource, or a chunk of memory, and he's got to give it back when he's done. This doesn't happen on anything on the Commodore 64 or IBM PC, so it must be a Bad Thing.

Many of these things aren't all that bad, and some are actually good. Once you get used to it, the Amiga CLI wild card system is much more powerful than those that allow only the standard * and ? patterns. The management of resources is necessary to multi-tasking, and once a programmer discovers how nice the shared library system is, or how useful it can be to have several processes share a single file, there "shortcomings" become "features". They're simply part of one's adjustment to the Amiga OS.

There are, however, a few things many users and programmers plan to never adjust to; these are in some cases real Amiga shortcomings. And whenever someone complains about the Amiga, they usually mutter something about the DOS and slow directory listings in the same breath.

AmigaDOS or Awful DOS?

The AmigaDOS disk operating subsystem of the Amiga OS is most maligned part of the Amiga operating system. It's slow, it's poorly documented, and it's written in an alien BCPL language that horrifies BASIC, C, Modula, Fortran, and LISP programmers alike.

The typical user is angered every time they type the 'Dir' command from CLI or opens a Workbench window. The same thing happens when wildcards are given to CLI commands, or a directory window is opened in most commercial programs. The CLI doesn't have any "built-in" commands like the MS-DOS CLI, which means that each and every command must be loaded from disk. These things are slow, right, so the DOS must be a Bad Thing.

Actually, AmigaDOS isn't a bad thing at all. It's a rather powerful system, much more flexible than most disk operating systems on personal computers today. AmigaDOS makes it quite easy to add in additional devices, and these devices can be added in at various logical levels. Everyone is familiar, of course, with the standard 3 1/2" floppy drives, and a few are discovering things like 5 1/4" floppy drives can be added very easily and be fully supported by all DOS commands. The same is true for hard disk drives and RAM disks that many people use today.

But it is also true for future things, like laser disk drives and networks. Someday you may have a bunch of Amigas all operating over a local area network to some central machine that actually supports the disk everyone is using. But for the user, aside from perhaps typing CD NET: instead of CD DF0:, all your commands and software will operate just as it does for your floppy or hard disk. (By the way, something like this NET: example actually does exist already, and it works as I described.)

Handlers and devices

Two levels of DOS data abstraction make this all possible: the "device" and the "handler". A device is a program closely tied to a particular piece of hardware. The floppies are managed by a program called "trackdisk.device"; a hard disk is managed by a similar program called "harddisk.device". A typical device supports low level commands; simple operations such as "write a block of data", "read a block of data", "format a track", etc.

All devices are accessed via the EXEC's Standard I/O library functions: to talk to one, the programmer must create a standard I/O request structure and then call OpenDevice() to open the device, which could including loading the device program from the system disk. Device commands are then placed in the I/O request structure and sent to the device program via the DoIO() function, for synchronous I/O requests, or the SendIO() function, for asynchronous I/O requests.

Yes, I said asynchronous! In most cases the I/O routine is a separate task which can be commanded while your program does other things instead of waiting. Devices, being managed by the Amiga's EXEC, are below the DOS system level and as such have no knowledge of any file structure; their main purpose is to provide a consistent interface to system hardware.

The problem with directly accessing a device is that its difficult to impossible to talk to different devices in a device-independent way. I normally talk to things called "DF0:", "RAM:", and "DH0:". These are all DOS level objects, some of which may be associated with devices, some of which probably don't have any direct device driver. The DOS object associated with the DOS level "DF0:" or whatever is called a handler.

A handler is just another program (a process, specifically), managed at the DOS level. Most file-oriented devices, including all floppies, many hard disks, and even some RAM disks, are driven by a handler called "File System". Handlers take a larger and higher level command set defined by DOS; actions such as Open, Read, Write, Seek, and Close on a file; CreateDir, Parent, and CopyDir operation on a directory, and various other informational and management actions. All handlers, regardless of whether they support files or not, are accessed in a consistent manner, no matter what the underlying device may actually be.

All communication to handlers is based on EXEC messages, in a well defined format call a DOS packet. A packet is a data structure that supplies command information to a handler, similar in purpose to the I/O request structure used by devices. A packet is sent to a device handler message

continued...

port via EXEC message calls. Similarly to devices, DOS packets can be synchronous or asynchronous. Handlers, however, don't normally have any knowledge of the underlying device, unless they're written to support one specific device. And for all but one packet type, they never have very low level access to any specific device.

So where is the slowdown? The largest part of the slowdown is in the standard DOS "File System". The File System is a general purpose DOS handler, written to drive the floppies via the trackdisk.device, but general enough to be used for many things other that floppies. Since it's much easier to write a device driver than a handler, as there are many fewer commands to implement in a device, there have been many devices written for the standard File System handler.

Using the standard handler, it is also much easier to make sure that your new device works just like other devices, so you don't run into compatibility problems. As long as my harddisk.device or RAMdisk.device follows the same command format as the trackdisk.device, I'm free to use the File System handler for my new hardware. So all I worry about is the simple stuff like reading, writing, and formatting my new thing, while the File System handler handles high level things like the implementation of files and directories. Unfortunately, it is this standard File System's directory implementation that is causing the slowness, based on several properties of the system.

Physical disk layout

Like most computer disks, an Amiga disk is organized into circular regions, called cylinders or tracks. Each cylinder is one full rotation of the disk. An Amiga 3 1/2" floppy disk supports 80 cylinders. Since the drive has a read/write head on each side of the disk, the disk effectively has 160 total physical tracks. Each track is subdivided into sections of equal size, called sectors. Floppies support 11 sectors per track, thus resulting in a total disk content of 1760 sectors. (Sectors are also known as blocks.) A hard disk will usually have more read/write heads, more cylinders, and thus more sectors than a floppy disk. A sector is the smallest division of the Amiga's disk format. Most disk devices deal with sectors directly.

The File System always asks for things by sector number. For any device it serves, the File System knows only the starting and ending block numbers. It knows absolutely nothing about the physical organization of the disk. The disk device is the glue that takes the language a physical disk speaks, like the addressing by cylinder, head, and sector, and converts it to the standard logical File System method of addressing a disk as a series of uniformly sized continuously numbered data blocks. The File System decides what to put into these blocks. It is actually possible for several alternate File Systems to use the same device driver, just as it is possible for a single File System to use several different device drivers.

Logical block structure

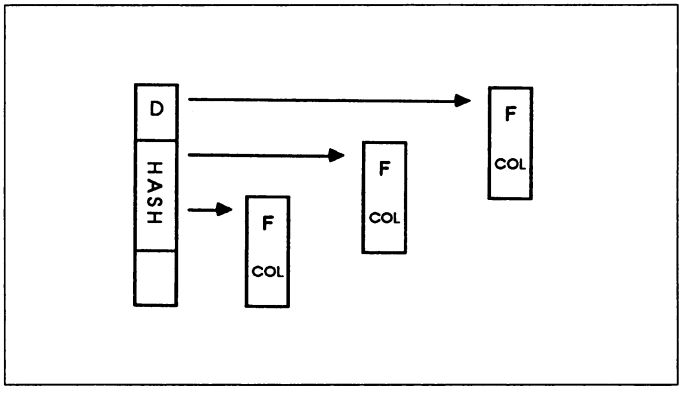
The standard File System imposes a specific block format on each block. The standard size of each block is 512 bytes, addressed as a series of 128 long, 32-bit words. The first six longwords of every block are used by the Handler for bookkeeping. The first longword contains the type of the block, which is either a SHORT_FILE, which can be a file or directory, or a DATA_BLOCK, which is a piece of a regular file. The next several longwords contain things like block backpointers (good for error recovery), lengths, file block counts, and finally the block's checksum, in the sixth longword. If the block is a DATA_BLOCK, the next 122 longwords, or 488 bytes, will be file data.

For SHORT_FILE blocks, there is more standard data at the end of a block. The last longword contains the block's sub type. Such a block can be either a ROOT_DIRectory, a USER_DIRectory, or a FILE. Directory commands are of course mainly concerned with directory blocks of some kind, so that is what is examined here.

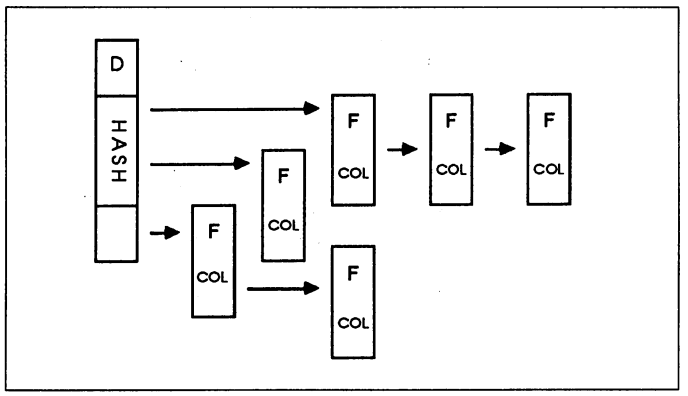
The File System stores a hash table of block pointers (also called KEYs), 72 longwords in length, in each directory block, starting right after the checksum. Each non-zero pointer in such a table is the disk block address of a file or another directory within the directory being examined. The "hashing" is accomplished via a function that operates on file names to produce a 1 out of 72 index into the table from the string. The hashing function used by AmigaDOS can be represented in C as:

```
/* Hash function example */

#define HASHSIZ 72      /* Fixed size of hash table */

ULONG hash(s)
unsigned char *s;
{
  ULONG res;
  unsigned short c;

  res = strlen(s);
  while (*s) {
    c = *s++;
    if (c >= 'a' && c <= 'z')
       c = c - 'a' + 'A';
    res = ((res * 13 + c) & 0x7ff);
  }
  return (res % HASHSIZ);
}

/* End of example */
```

As each file is added to a directory, the hash number for that file is calculated, and the key allocated for the file header block will be inserted in the list at the hashed position. Obviously, this can't go on forever; soon a file will hash to a value previously used. In this case, the collision field found in each file header and directory header can hold the key of the colliding file. Of course, now, instead of a nice, relatively clean directory list that looks like this:



There will be collision chains that build up, so that the structure of a directory will look more like:



On the average, a directory would need a lot of file entries to accumulate more than just a few collision chain links. Obviously, a directory needs more than 72 files to insure that hash chains are used, but it is possible to have a hash chain with just two entries. In reality, small directories tend to have no chains, while large directories have of course more. It is rare to have directories with more than a hundred entries, so the hashing chains are usually short.

Directories, slow and fast

The Amiga directory organization above has some definite advantages, but it is also the cause of the 'dir' slowness. Disk drives are slowest at stepping from track to track. Therefore, this accounts for most of the waiting.

Finding a named file is very fast. If I give a file name to a Handler function that will find that file, as the Open() function does, then the disk seeks the directory header, hashes the file name, seeks the key from the hash table, does one string comparison, and at worst, maybe do another seek and compare or two if there are collision blocks.

Consider a wildcard search. This happens in a directory listing. Here, there is one seek for the directory header, one seek for each hash table entry, and one seek for each collision block. That's a seek for every file in the directory, plus one. The DOS directly supports these operations via function calls or packets: Examine() or ACTION_EXAMINE_OBJECT can perform the required seek for the directory header, ExNext() or ACTION_EXAMINE_NEXT can look up the next file in a directory, via the hash table and any collision keys.

AmigaDOS CLI commands like 'dir' and 'list' use these same handler packets to get directory listings. Note these packets don't imply any particular disk structure. In the case of the standard File Handler, there is a potential seek for each entry. An alternate file system such as RAM: disk provides the same packet types, but may do something completely different in order to fulfill their requested actions.

Unfortunately, these packet actions are a big part of the slow directory problem you get with the standard File Handler. When the FileHandler is asked to supply the next file in a directory, via an EXAMINE_NEXT packet, it supplies the next block found in the hash table or a collision block, based on hash table ordering and any collision blocks encountered.

The order of the hash table, though, has nothing at all to do with the ordering of physical disk blocks out on the disk. In the worst case, this means the 'dir' program may read a track for the first block, then scoot across the disk for the next, and come back to the first track for the next block. Obviously, there must be a better way!

A better way

The better way I've found is the FDir program. FDir reads a directory header and sorts the hash table keys to optimize drive head movement. In this way, once the drive head is at a particular track, all of the blocks on that track are read read before moving to the next track. Adjacent tracks will be processed as the head moves across the disk.

In order to do this, though, the normal EXAMINE packets can't be used. They are the only normal way to query a handler's disk structure for files in a directory. Each DOS handler localizes its specific information, as mentioned, thus allowing all DOS programs to work generically on any handler.

Thus, FDir must use special knowledge of the file structure to perform its sorting operations. First, it checks to see if the device can work with the File System. Devices based on 128 longword blocks can use the enhanced algorithm, otherwise FDir falls back on conventional EXAMINE loop, much as 'dir' and 'list' use.

Some device structures hold the driver and device unit number used by a particular DOS object. With this information, FDir can read raw blocks with standard EXEC I/O functions and a proper I/O request block. After all, it is the device's job to read raw blocks by block number from a physical device. However, this is more work than necessary.

Earlier, I hinted at a DOS packet that does not isolate the caller from the underlying device. This packet is called the ACTION_GET_BLOCK packet. Given a block number and a block-sized buffer, this packet reads a specific block from a device.

This method is better because it happens at the packet level, with DOS packets sent direct to a device's process ID, or PID. (The PID is actually the address of the message port of the handler process for that volume),

In this case, packets and PIDs are simpler to use when dealing with abstract devices. As a handler should, they supply all driver-specific information for you. Using this ACTION_GET_BLOCK packet, FDir will next read the given directory header block, and check its type. If the type is wrong, either it's not a directory or not using the standard File System; in either case, the fall-back EXAMINE-based routine is called.

Should it be the correct type, the directory block's hash table is scanned for valid keys, which are inserted into a sorted list of keys. The directory listing now begins. Each block, in physical order, will be read and displayed. Additionally, any collision blocks found are inserted back into the sorted list. This collision key insertion is done in an intelligent fashion. The direction of head movement is considered; if the key is further on in the direction we're currently moving, the key is inserted at the appropriate place near the head of the list. Should the key be in the other direction, it will be inserted near the back of the list to be picked up on the way back. In this way, the heads sweep back and forth, reading all possible blocks in each pass, which greatly reduces the jumping back and forth of the EXAMINE method.

One final device-specific enhancement is used to speed things up. When the device-specific information is considered at the start of the program, the size of the device's track or cylinder (via an option) is noted. During the fast listing loop, if a collision key is found that's anywhere on the current track (or cylinder), it's processed right away, instead of being placed in the key list at all. For devices like the floppy disk, which do full track reads all the time, this system can be a very good speedup.

continued...

Benchmarks: Your mileage may vary

Before I get into the details of how I actually coded this system, I've got a list of benchmarks that should be of interest. Table 1 compares the speeds of FDir and 'dir' and 'list'. The FDir command accepts two switches. The "-c" switch causes the program to check if a collision key is anywhere (accessible by any head) on the current cylinder; normally the program only checks the current track.

Floppies seem to work just a bit better only considering tracks, which is why I left track-only checking as the default. The other switch the program accepts is the "-n" switch, which forces the program into its EXAMINE loop fall-back mode, regardless of the type of File Handler involved. This allows very close benchmarking of the sorting and optimizing methods, since almost everything else is done the same in the the Fast and Normal directory methods.

TABLE 1: Timing of various directory commands (Seconds)

F#	P#	FDir		FDir -c		FDir -n		Dir		List	
10	20	1.28	3.44	1.27	3.49	1.23	4.05	1.31	4.01	1.82	4.56
50	10	4.88	11.83	4.90	11.74	7.59	17.61	8.48	18.27	10.21	20.22
100	10	9.26	18.67	10.85	18.93	26.05	35.40	29.29	37.82	31.47	40.61
200	5	18.46	36.71	20.04	36.95	68.69	81.65	77.39	87.83	78.78	91.83
300	5	29.03	53.41	28.82	54.14	113.69	125.85	129.06	138.08	128.25	140.72
400	5	38.25	58.32	40.92	59.23	157.11	176.70	181.15	199.84	174.59	195.79
600	1	59.83	N/A	62.05	N/A	251.37	N/A	305.65	N/A	218.62	N/A
LENGTH:				7520				8128		8440	

Although all of these programs are very similar in size, I've run these benchmarks with all of them in the RAM: disk just to minimize the impact of the program length on performance. All of the benchmarks were created using two additional simple programs I wrote for testing. One of these programs repeats a CLI command any number of times, via the DOS Execute() function, and times the execution of that command (averaged over the number of passes specified) via the Amiga's timer.device.

The second program simply creates any number of small files. The first column lists the number of files on the disk during the test. In all cases, the disk was in DF1: of a standard PAL Amiga with two floppies and FAST expansion memory. No disk buffers were added to DF1: for any pass. The second column lists the number of runs the listed time was averaged over.

The test runs follow, organized as a pair of columns for each run on each file setup. The first of the pair represents the test run on a disk built in the normal order in which the standard File System handler allocates its new blocks. The second of the pair is the same number of files on a more munged disk, where blocks have been repeatedly added and deleted. The difference is that, using the normal allocation scheme, disk blocks tend to get allocated consecutively and locally, while on a badly munged disk the directory could possibly exist all over the disk, and will very likely be in a very random ordering. Both of these factors are expect to hurt the performance of both directory schemes.

Note that all of the times in the table are in seconds, and that the test with 600 files in the directory was never run with a munged directory. This was intentionally skipped, because the performance of the overall file system drops so low when you've got around 600 files in a single directory that it takes far too much time to create and delete files. The length of each program, in bytes, is shown at the end of the table, for comparison purposes. I don't think the length differences amounted to a significant difference in run times.

This table shows that in most cases the sorted scheme used by FDir is faster than the standard EXAMINE scheme supported by DOS. In the case of a very small directory, the FDir version of the EXAMINE method proved to be just a bit faster, and this will probably also be true in certain cases on RAM and Hard disks that use the standard File Handler. The sorting scheme used by FDir has a finite overhead versus the EXAMINE method, though for all but the smallest, cleanest directories, that's quickly overcome.

The other conclusions from this are that, at least for floppy disks, a same-track check, as performed for hash chain collisions under the normal FDir, is usually more efficient that the same-cylinder check available via the "-c" option. It is also interesting to note that the overhead in factors other than head movement becomes apparent, especially with the supplied 'dir' and 'list' commands. List starts out taking the longest, which is most likely due to the extra output it produces listing its information for each file on a single line. However, with somewhere between 200 and 400 files to list, Dir's alphabetic sorting becomes significant, and it becomes the slowest command. FDir, listing three files per line, unsorted, is the fastest by at least a shade in every case, regardless of options.

Next time I'll have listings and more expplanations ...so stay tuned.

•AC•

Modula-2 Programming
on the Amiga™

RAW Console Device Events

by Steve Faiwiszewski

AmigaDOS can inform a program of various events (such as arrow and function key presses, mouse clicks, etc) through the use of the RAW (as opposed to the CON) console device. This information is conveyed by sending special character sequences to the program's RAW window. Theoretically, one should be able to use the routines in the InOut module to receive these character sequences from the RAW window. However, for better or for worse, TDI decided that the standard IO module should filter those character sequences, and just pass on an Escape if one of them is received.

Presented here is a general purpose module called MyRawInOut, which will enable you to access RAW events. Based on the original InOut module supplied by TDI, MyRawInOut provides procedures for opening and closing a window, and for reading from and writing to it.

Also included are two programs which make use of MyRawInOut. The first, called RawDemo, demonstrates how to activate the various types of RAW events and displays them in their raw format. The second program, RawDemo2, is an example of how to parse the RAW events in order to detect arrow and function key selection.

RAW Arrow/Function Key Events

When a RAW window is first opened (i.e. in its default state), the function and arrow keys generate the following sequences:

	Unshifted	Shifted
F1	<CSI>0~	<CSI>10~
F2	<CSI>1~	<CSI>11~
F3	<CSI>2~	<CSI>12~
F4	<CSI>3~	<CSI>13~
F5	<CSI>4~	<CSI>14~
F6	<CSI>5~	<CSI>15~
F7	<CSI>6~	<CSI>16~
F8	<CSI>7~	<CSI>17~
F9	<CSI>8~	<CSI>18~
F10	<CSI>9~	<CSI>19~
Help	<CSI>?~	<CSI>?~ (same)
Up	<CSI>A	<CSI>T
Down	<CSI>B	<CSI>S
Left	<CSI>C	<CSI> A
Right	<CSI>D	<CSI> @

CSI is a one-byte code of 9B (hex). It stands for Control Sequence Introducer. All other keys generate their normal key codes (i.e. 'A' generates 65, '0' generates 48, etc.). The RawDemo2 program parses these escape sequences according to the above table, and prints whether the pressed key is a function key, an arrow key, or simply a "normal" key.

Other RAW Input Events

The RAW console device can supply even more information. Events such as any key press, a mouse button click, window activation, and disk insertion can all be detected through RAW escape sequences. To activate the reporting of these various events all that has to be done is to send to the RAW window an escape sequence of the following format: <CSI>x{ where x is a number from 1 to 16, according to the following table (from the AmigaDOS Developer's Manual):

Event Number	*Description*
0	no-op
1	RAW keyboard input
2	RAW mouse input
3	Window activated event
4	Pointer position
5	(unused)
6	Timer
7	Gadget pressed
8	Gadget released
9	Requester activity
10	Menu numbers
11	Close gadget
12	Window resized
13	Window refreshed
14	Preferences changed (not implemented)
15	Disk removed
16	Disk inserted

The RawDemo program will allow you to selectively activate the reporting of RAW events, and to then experiment with those events. When running the program, you will select which events should be reported by entering the event number. To keep things simple RawDemo looks for just one digit, so to select events 10 through 16 use letters 'A' through 'G'. Be aware that if you select RAW keyboard input (event #1), the Esc key - which is used by RawDemo to signal it to exit - will stop generating a single Escape character, and the only way to exit from the program is to use the Break command and to send RawDemo a Ctrl-C signal.

Notice how the program uses the MoreCharsComing procedure to determine what characters should be grouped together on one display line. If you have more than one character coming in within a very short time interval (say 50 milliseconds) you know that these characters belong to an escape sequence.

By the way, RawDemo also demonstrates how to detect Control-C signals: all you have to do is obtain a pointer to your own task record (a TaskPtr) using the FindTask procedure, and then just test the tcSigRecvd field in that record for the presence of SIGBreakC.

About Run-time Error Handling

During the development of RawDemo I ran into some run-time errors; most were due to my own bugs, but one occurred because the TDI compiler couldn't handle the size of one of my procedures (I finally got around that problem by breaking that procedure down to smaller ones - a good thing to do anyway).

The previous release of the compiler would crash the Amiga when a run-time error was encountered, but programs compiled using version 3.00 just return to the CLI. This is all nice and good, but the RAW window which my program opened remained opened after the program exited, and the only way to get rid of it was to reboot the machine. Using the post-mortem debugger (otherwise known as the PMD) didn't help either, since the debugger doesn't really know anything about any screens, windows, files or any other resources a program might allocate. What I had to do was to trap the run-time error myself and close my window if an error occurred. But how do I trap the error so I could close things down gracefully and still

continued...

get to call the PMD? Well, the solution was quite simple: Declare a variable of type PROC (I have one called OldErrorProcessor), which is a procedure variable. Next, assign to it the value of the ErrorProcessor procedure variable (imported from the AMIGAX module). Then, point ErrorProcessor to your error handling procedure (ErrorTrapper in my case). Within your error handler procedure you can do whatever you have to, as long as the last thing you do is call your PROC variable (OldErrorProcessor in RawDemo). This will cause the PMD to execute once you closed down things gracefully. Oh, one more thing: ALWAYS ALWAYS import Trapper! You never know when you'll need it, and it's pretty small anyway. Until next time, keep on modulating.

Listing One

```
DEFINITION MODULE MyRawInOut;

(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * )
(* Provides routines to access all RAW keyboard events     *)
(*  Created by Steve Faiwiszewski, May 1987.               *)
(*  module.                                                *)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * )

PROCEDURE MyOpen(VAR FileName: ARRAY OF CHAR);
(* Open a new file to read from and write to *)

PROCEDURE MyClose;

PROCEDURE MoreCharsComing(TimeInterval : LONGCARD) : BOOLEAN;

PROCEDURE MyRead(VAR ch: CHAR);

PROCEDURE MyWriteCard(number: CARDINAL);
(* Do a simple print of a cardinal.              *)
(* Leading blanks capability is not implemented *)

PROCEDURE MyWriteHex(number: CARDINAL);

PROCEDURE MyWrite(c : CHAR);

PROCEDURE MyWriteLn;

PROCEDURE MyWriteString(VAR line : ARRAY OF CHAR);

END MyRawInOut.
```

Listing Two

```
IMPLEMENTATION MODULE MyRawInOut;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
(*  Routines to access all RAW keyboard events.                 *)
(*  Created by Steve Faiwiszewski, May 1987.                    *)
(*                                                              *)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)

FROM Libraries   IMPORT OpenLibrary;
FROM DOSLibrary  IMPORT DOSBase, DOSName;
FROM Conversions IMPORT ConvertToString;
FROM Streams     IMPORT STREAM, Connect, Disconnect,
                        ReadChar, WriteChar,WriteString;
FROM DOSFiles    IMPORT Open, Close, WaitForChar,
                        ModeNewFile, FileHandle;


CONST
    LF = 12C;

VAR
   MyFile : STREAM;

PROCEDURE MyOpen(VAR FileName: ARRAY OF CHAR);
(* Open a new file to read from and write to *)
VAR
    fh: FileHandle;
BEGIN
    fh := Open(FileName,ModeNewFile);
    IF fh <> 0 THEN
      Connect(MyFile,fh);
    END;
END MyOpen;

PROCEDURE MyClose;
(* close out file *)
VAR
    fh: FileHandle;
BEGIN
    fh := MyFile.handle;
    IF fh > 0 THEN
      Disconnect(MyFile);
      Close(fh)
    END;
END MyClose;

PROCEDURE MoreCharsComing(TimeInterval : LONGCARD) : BOOLEAN;
(* see if there are any more chars coming in withing the *)
(* time interval supplied.                               *)
BEGIN
    RETURN WaitForChar(MyFile.handle,TimeInterval)
END MoreCharsComing;

PROCEDURE MyRead(VAR ch: CHAR);
BEGIN
    ReadChar(MyFile,ch);
END MyRead;

PROCEDURE MyWriteCard(number: CARDINAL);
(* Do a simple print of a cardinal.              *)
(* Leading blanks capability is not implemented *)
VAR string: ARRAY [0..39] OF CHAR;
    dummy: BOOLEAN;
BEGIN
    ConvertToString(LONGCARD(number),10,FALSE,string,dummy);
    WriteString(MyFile,string);
END MyWriteCard;

PROCEDURE MyWriteHex(number: CARDINAL);
(* Do a simple print of a cardinal.              *)
(* Leading blanks capability is not implemented *)
VAR string: ARRAY [0..39] OF CHAR;
    dummy: BOOLEAN;
BEGIN
    ConvertToString(LONGCARD(number),16,FALSE,string,dummy);
    WriteString(MyFile,string);
END MyWriteHex;

PROCEDURE MyWrite(c : CHAR);
BEGIN
    WriteChar(MyFile,c)
END MyWrite;

PROCEDURE MyWriteLn;
BEGIN
    WriteChar(MyFile,LF)
END MyWriteLn;

PROCEDURE MyWriteString(VAR line : ARRAY OF CHAR);
BEGIN
    WriteString(MyFile,line)
END MyWriteString;

BEGIN
    IF DOSBase = 0 THEN
       DOSBase := OpenLibrary(DOSName,0);
    END;
END MyRawInOut.
```

Listing Three

```
MODULE RawDemo;
        (* Demonstration of RAW console device events. *)
        (* Created by Steve Faiwiszewski, May 1987.    *)

FROM MyRawInOut  IMPORT MyOpen, MyClose, MyRead, MyWrite,
                  MyWriteString, MyWriteLn, MoreCharsComing;
FROM DOSLibrary  IMPORT SIGBreakC;
FROM Tasks       IMPORT TaskPtr, FindTask, MyTask;

FROM SYSTEM      IMPORT BYTE;
FROM AMIGAX      IMPORT ErrorProcessor;
IMPORT Trapper;

CONST
    BELL = 07C;
    FF = 14C;
    ESC = 33C;
    CSI = 233C;
    TimeInterval = 50;

TYPE
    CharSet = SET OF CHAR;

VAR
    OldErrorProcessor : PROC;
    Myself  : TaskPtr;

PROCEDURE CtrlC(): BOOLEAN;
(* see if control-c signal has arrived *)
BEGIN
    RETURN SIGBreakC IN Myself^.tcSigRecvd
END CtrlC;


PROCEDURE LoopAround;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
(* Wait for chars from the RAW window, & display them when *)
(* received.                                               *)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

VAR
    c   : CHAR;
BEGIN
    REPEAT
        REPEAT
            MyRead(c);
            IF c = CSI THEN
                MyWriteString('<CSI>')
            ELSIF c = ESC THEN
                MyWriteString('<Esc>')
            ELSE
                MyWrite(c)
            END;
        UNTIL NOT MoreCharsComing(TimeInterval) OR CtrlC();
        MyWriteLn
    UNTIL (c = ESC) OR CtrlC();
END LoopAround;

PROCEDURE Confirm(): BOOLEAN;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
(* Be sure user wants to activate RAW keyboard input event *)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
VAR
    c : CHAR;
    done : BOOLEAN;
BEGIN
    MyWriteLn;
    MyWriteString("You won't be able to exit using Esc key!
Confirm? (Y/N) ");
    REPEAT
        MyRead(c);
        c := CAP(c);
        done := (c = 'N') OR (c = 'Y');
        IF NOT done THEN MyWrite(BELL) END;
    UNTIL done;
    IF c = 'N' THEN
```

continued...

```
      MyWriteString('No')
   ELSE
      MyWriteString('Yes'); MyWriteLn;
      MyWriteString('The Break command should still work
(e.g. Break 1 C)')
   END;
   RETURN (c = 'Y')
END Confirm;

PROCEDURE GetInput(VAR chars : ARRAY OF BOOLEAN): BOOLEAN;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
(* Get the input from the user.                           *)
(* To keep things simple, input one character for each of *)
(* the 16 possible RAW events. Use characters '1' thru '9' *)
(* and 'A' thru 'G'.                                      *)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
VAR c : CHAR;
    index,
    base : CARDINAL;
BEGIN
    LOOP
       MyWriteString("Enter character 1..9 and A..G ('0' to
proceed): ");
       MyRead(c);
       IF c = ESC THEN
           RETURN FALSE
       ELSIF c = '0' THEN
           MyWrite(c);
           RETURN TRUE
       END;
       c := CAP(c);
       IF c IN CharSet{'1'..'9','A'..'G'} THEN
          MyWrite(c);
          IF c < 'A' THEN
             base := ORD('0')
          ELSE
             base := ORD('A') - 10
          END;
          index := ORD(c) - base - 1;
          IF chars[index] THEN
             MyWriteString('  (already selected)')
          ELSE
             IF index = 0 THEN
                 chars[index] := Confirm()
             ELSE
                 chars[index] := TRUE;
             END
          END
       ELSE
          MyWrite(BELL)
       END;
       MyWriteLn;
    END; (* loop *)
END GetInput;

PROCEDURE SelectRAWevents(): BOOLEAN;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
(* Get selection of various RAW events to be activated,  *)
(* and activate them.                                     *)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
VAR
    index : CARDINAL;
    chars : ARRAY[1..16] OF BOOLEAN;
    string: ARRAY[0..5] OF CHAR;
BEGIN
    FOR index := 1 TO 16 DO chars[index] := FALSE END;
    IF NOT GetInput(chars) THEN RETURN FALSE END;
    FOR index := 1 TO 16 DO
        IF chars[index] THEN
            IF index > 9 THEN
                string[0] := '1';
                string[1] := CHR(ORD('0')+index-10);
                string[2] := 0C;
            ELSE
                string[0] := CHR(ORD('0')+index);
                string[1] := 0C
            END;
            MyWrite(CSI); MyWriteString(string); MyWrite('{')
        END
    END;
    MyWrite(FF);
    MyWriteString('RAW event selection completed.');
MyWriteLn;
    IF chars[1] THEN
       MyWriteString('Use the Break command')
    ELSE
       MyWriteString('Press Esc');
    END;
    MyWriteString(' to exit.'); MyWriteLn;
    RETURN TRUE
END SelectRAWevents;

PROCEDURE ErrorTrapper;
(* Close our window, and call TDI's error handler *)
BEGIN
    MyClose;
    OldErrorProcessor;
END ErrorTrapper;

BEGIN
    Myself := FindTask(0);
    MyOpen('RAW:0/0/640/200/TDI M2 Raw Demo.');
    OldErrorProcessor := ErrorProcessor;
    ErrorProcessor := ErrorTrapper;
    MyWriteString('Normally pressing Esc will exit the
program');
    MyWriteLn;
    IF SelectRAWevents() THEN
       LoopAround;
    END;
    MyClose;
END RawDemo.
```

Listing Four

```
MODULE RawDemo2;
          (* * * * * * * * * * * * * * * * * * * * * * * *)
          (* Demonstration of RAW console device arrow & *)
          (* function key events.                        *)
          (* Created by Steve Faiwiszewski, May 1987.    *)
          (*       Plink : THE INTERN                    *)
          (*       CIS   : 74106,425                     *)
          (* * * * * * * * * * * * * * * * * * * * * * * *)

FROM MyRawInOut  IMPORT MyOpen, MyClose, MyRead, MyWrite,
MyWriteString,
                        MyWriteCard, MyWriteHex, MyWriteLn;
FROM SYSTEM      IMPORT BYTE;
IMPORT Trapper;

CONST
    ESC = 33C;
    CSI = 233C;

TYPE
    KeyClass =
(NormalKey,ArrowKey,FunctionKey,ShiftFunctionKey,HelpKey);
    ArrowType =
(Up,Down,Left,Right,ShiftUp,ShiftDown,ShiftLeft,ShiftRight);
    CharSet = SET OF CHAR;


(* ++++++++++++++++++++++++++++++++++++++++++++++++ *)
PROCEDURE SpecialRead(VAR key : KeyClass; VAR value : BYTE);

VAR
    c : CHAR;
    i : CARDINAL;
    CharTable : ARRAY[0..4] OF CHAR;
    done : BOOLEAN;
BEGIN
    MyRead(c);
    IF c <> CSI THEN
        key := NormalKey;
        value := BYTE(c);
    ELSE
        i := 0;
```

```
        REPEAT
           MyRead(c);
           done := c IN
CharSet{'@','A','B','C','D','S','T','~'};
           CharTable[i] := c;
           INC(i)
        UNTIL done;
        IF i = 1 THEN (* arrow keys *)
           key := ArrowKey;
           CASE CharTable[0] OF
              'A' : value := BYTE(Up)          |
              'T' : value := BYTE(ShiftUp)     |
              'B' : value := BYTE(Down)        |
              'S' : value := BYTE(ShiftDown)   |
              'C' : value := BYTE(Left)        |
              'D' : value := BYTE(Right)
              ELSE
                   MyWriteString('Program Error 1')
           END;
        ELSIF i = 2 THEN
           IF CharTable[0] = ' ' THEN
              key := ArrowKey;
              IF CharTable[1] = 'A' THEN
                 value := BYTE(ShiftLeft)
              ELSE
                 value := BYTE(ShiftRight)
              END
           ELSIF CharTable[0] = '?' THEN
              key := HelpKey
           ELSE
              key := FunctionKey;
              value := BYTE(1 + ORD(CharTable[0]) -
ORD('0'));
           END
        ELSIF i = 3 THEN
           IF CharTable[0] = ' ' THEN
              key := ArrowKey;
              IF CharTable[1] = 'A' THEN
                 value := BYTE(ShiftLeft)
              ELSE
                 value := BYTE(ShiftRight)
              END;
           ELSE
              key := ShiftFunctionKey;
              value := BYTE(1 + ORD(CharTable[1]) -
ORD('0'));
           END
        ELSE
           MyWriteString('Program Error 2')
        END
    END
END SpecialRead;

PROCEDURE DoFunctionKey(c : CHAR);
BEGIN
    MyWriteString('Function key ');
    MyWriteCard(ORD(c));
END DoFunctionKey;

PROCEDURE DoShiftFunctionKey(c : CHAR);
BEGIN
    MyWriteString('Shift Function key ');
    MyWriteCard(ORD(c));
END DoShiftFunctionKey;

PROCEDURE DoArrowKey(c : CHAR);
BEGIN
    CASE ArrowType(c) OF
        Up        : MyWriteString('Up')          |
        Down      : MyWriteString('Down')        |
        Left      : MyWriteString('Left')        |
        Right     : MyWriteString('Right')       |
        ShiftUp   : MyWriteString('Shift Up')    |
        ShiftDown : MyWriteString('Shift Down')  |
        ShiftLeft : MyWriteString('Shift Left')  |
        ShiftRight: MyWriteString('Shift Right')
        ELSE
                  MyWriteString('Program Error 3')
    END;
```



```
    MyWriteString('Arrow key');
END DoArrowKey;

PROCEDURE DoNormalKey(c : CHAR);
BEGIN
    MyWriteString('key: ');
    MyWrite(c);
    MyWriteString(' (Decimal: ');
    MyWriteCard(ORD(c));
    MyWriteString(',  Hex: ');
    MyWriteHex(ORD(c));
    MyWrite(')');
END DoNormalKey;

PROCEDURE LoopAround;
VAR
    c   : CHAR;
    Key : KeyClass;
BEGIN
    REPEAT
        SpecialRead(Key,c);
        CASE Key OF
           HelpKey:          MyWriteString('Help key')   |
           FunctionKey:      DoFunctionKey(c)            |
           ShiftFunctionKey: DoShiftFunctionKey(c)       |
           ArrowKey:         DoArrowKey(c)               |
           NormalKey :       DoNormalKey(c)
        END;
        MyWriteLn
    UNTIL c = ESC;
END LoopAround;

BEGIN
    MyOpen('RAW:0/0/640/200/TDI M2 Function & Arrow keys
Demo.');
    LoopAround;
    MyClose;
END RawDemo2.
```

•AC•

AmigaBASIC™ Patterns

Line patterns, fill patterns and multicolored patterns are readily available to all AmigaBASIC programmers.

by Bryan Catley

One of the more unique features of AmigaBasic is the ability to draw and paint in patterns. This capability can produce startling results. So, why do so few AmigaBasic programs actually use this capability?

Well, there are probably a number of reasons, but one of the more obvious ones is that we must design the patterns using binary digits and specify them in hexadecimal digits. (They could also be specified in decimal, but that would be even more complicated!) Now, if you know little about binary and hexadecimal, and are wondering if you should read any further, please don't give up yet. This article is NOT a lecture on numbering systems (I promise!)- - - and the results you can achieve by using patterns will be well worth the time spent studying this article.

There are two simple rules to remember about patterns: a). they are designed using 0's and 1's (0's are drawn in the background color, while 1's are drawn in the foreground color - except with multicolored patterns) b). Patterns are always 16 bits, pixels, positions or whatever wide and they repeat as often as necessary. Further, there are three distinct kinds of patterns: Line Patterns, Fill Patterns and Multicolor Patterns. Let's take a look at each type.

Line Patterns

It may not be too obvious, but lines are always drawn using patterns! The solid default pattern masks the individual line pattern. Thus, the 16 bit pattern for a solid line may be represented as 1111111111111111. Now, if the pattern is repeated for the length of the line, we have a continuous solid line. Suppose we wanted a dashed line, composed of one pixel on, one off, etc. This pattern could be defined as 0101010101010101. What about a line of long dashes? 0111011101110111 would do the trick. Finally, a line composed of alternating short and long dashes might be defined as 0110111101101111. The trick lies in telling Basic about the pattern we want to use!

AmigaBasic provides a PATTERN statement which allows us to tell BASIC which pattern to use. However, the pattern is usually specified in hexadecimal! (Specification is in hexadecimal only because decimal is even more complicated to specify!). Thus, we must now convert those 0's and 1's to hexadecimal digits! As it turns out, this conversion is not difficult at all, but since we're not delving into numbering systems, you'll have to take my word for it!

The fact of the matter is that each set of four 0's and 1's may assume any of 15 combinations, and may be represented by a single hexadecimal digit. This arrangement means there must be at least 15 hexadecimal digits. Right? Yes, and 15 only. They are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Note how digits 10 through 15 are represented by the letters A-F. Having discovered this fact, we are now in a position to construct the following conversion table:

Binary	Hex	Dec *	Binary	Hex	Dec *
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

** For now, don't worry about the decimal column. You'll only need these figures for multicolored patterns.*

All this technicality means we can now take our 16 bit pattern, break it into four sets of four binary digits and then convert it to four hexadecimal digits. Thus, using the patterns described earlier, we have FFFF for the solid line, 5555 for the dotted line, 7777 for the line of long dashes and 6F6F for the line of alternating dashes.

All that's left is our use is the PATTERN statement telling Basic about our pattern. This process is accomplished via: PATTERN &Hnnnn where the &H tells Basic to expect some hexadecimal digits to follow, and the nnnn represents the four hexadecimal digits. Consider the following very short program (which you may care to try out, using several different patterns in the process):

```
COLOR 3,1
PATTERN &H6F6F
LINE (0,0)-STEP(160,160)
PATTERN &HFFFF
COLOR 1,0
```

First, we set our colors and pattern. We then draw a diagonal line and reset the pattern (to a solid line) and the colors. Resetting the pattern is very important, if you are using the Workbench screen for your program output! If you are using a custom screen, however, the pattern is automatically reset to solid when you close the screen (more accurately, the pattern terminates with the screen closing because each screen has its own pattern). Regardless, resetting is a good habit to develop.

Now, if we convert &HFFFF to decimal, we end up with -1, (not 15151515, please take my word for it), so we may also reset the pattern with PATTERN -1.

You should also know that once a line pattern has been set, it will automatically be used for drawing straight lines with the LINE statement, non-filled boxes with the LINE statement, the edges of a wedge removed from a circle (but not the circle itself) and any polygon drawn with the PolyDraw& operating system routine. There are more examples in Sample Program 1 which accompanies this article.

Fill Patterns

Once you understand line patterns (and they were easy, weren't they?), fill patterns can also become easy, even though they are a little more complicated. The reason for the extra complication rests in the fact that we now must define a pattern which covers an area, still 16 bits wide, and which may be of any depth up to that of the screen, provided it is a power or two (i.e. 2, 4, 8, 16, 32, etc).

Once the area pattern has been designed, we convert each 16 bit row to hexadecimal and define the rows in a series of DATA statements. These DATA statements are then read into a previously DIMensioned short integer array, which is then specified as the SECOND parameter of a PATTERN

statement. From this point on, all areas fills with LINE (using the bf option) AREAFILL or PAINT will be completed with the specified pattern. The pattern will be repeated as necessary to fill the area. Also, you should reset the fill pattern to solid before terminating the program.

All this confusion becomes a lot clearer when we look at the following example. Let's say we want a pattern to be made up of a series of small crosses. For this situation, we'll select eight lines for the pattern (remember, it must be a power of two), and design it as follows:

```
0000000000000000 =&H0000
0001100000011000 =&H1818
0001100000011000 =&H1818
0111111001111110 =&H7E7E
0001100000011000 =&H1818
0001100000011000 =&H1818
0000000000000000 =&H0000
0000000000000000 =&H0000
```

The code to do the actual drawing of this pattern follows, so why don't you try it out?

```
DIM Pat%(7)        'Dim Pattern Array
FOR n=0 TO 7
  READ x:Pat%(n)=x  'Fill with Pattern
NEXT
DATA &H0000, &H1818
DATA &H1818, &H7E7E
DATA &H1818, &H1818
DATA &H0000, &H0000
PATTERN ,Pat%    'Notice Comma
LINE (10,10)-(206,150),3,bf
FOR n=0 TO 7
  Pat%(n)=-1  'Fill with Solid Pattern
NEXT
PATTERN ,Pat%       'Restore Fill Pattern
```

Simple, isn't it? Now, there are three additional things you should know about fill patterns:

• a line pattern may be specified at the same time as a fill pattern, e.g. PATTERN &H6F6F,Pat%.

• once the PATTERN statement has been issued, Basic makes its own copy of the information stored in the named array, so you may ERASE the array if you are running short of memory. Don't forget, though, you will need to re-DIMension the array in order to return to the solid pattern.

• Very interesting effects may be achieved by filling the pattern array with random numbers!

Multicolored Patterns

So far, we have only discussed two color patterns where the pattern, (or 1 bits) is drawn in the specified foreground color, and the background (or 0 bits) is drawn in the current background color. However, at the cost of increased complexity, we can create patterns which use any or all of the available palettes!

The process is essentially the same: design the pattern, store it in an array and tell Basic about the pattern. However, because Basic has no built-in method of requesting a multicolored pattern, each step becomes a little more complex, as we have to fiddle with what we've got in order to achieve what we want!

Designing the pattern is, by far, the most complicated step. Besides specifying a pattern, we must also indicate the desired color for each pixel. Before we take a look at indicating color, let's review how the Amiga knows what color to display each pixel. This area is important to discuss because we can look at any Amiga display and think of it as one large multicolored pattern!

The maximum number of colors which may be used in any Amiga display is determined by the number of bit planes associated with the current screen. (In AmigaBasic, the depth parameter of the SCREEN statement specifies the number of bit planes to be used).

A bit plane is an area of memory where each bit represents one pixel on the screen. Now, if there is only one bit plane, and since each bit may only be a 0 or a 1, we are automatically limited to two colors. If the bit is a 0, it is drawn using palette 0. If it is a 1, the bit is drawn using palette 1. Simple. When we add a second bit plane, each pixel on the screen is now represented by two bits which may assume any of four values: 00, 01, 10 and 11. So, these bits are drawn using palettes 0, 1, 2 and 3, respectively. If we add a third bit plane, we end up with three bits per pixel for up to eight combinations, each directly indicating the palette to be used. We can keep adding bit planes until we have a total of four (for up to 16 colors) with a 640 pixel screen width, or even five (for up to 32 colors) with a 320 pixel screen width.

By now, it is probably obvious that the color palette used to draw each pixel on the screen depends directly on the combination of matching 0's and 1's in the bit planes. This technique is exactly the one we use when defining a multicolored pattern! We define the appropriate pattern for each available bit plane, ensuring the matching bits in each plane are a combination which will produce the desired color for that pixel. Unfortunately, this restriction also means we must learn a little bit (pun intended) of binary, in order to translate the desired palette numbers into bit combinations. To provide some help, a decimal column was given with the binary to hexadecimal table presented earlier. To get from 16 to 31, repeat the same 16 binary combinations, but start them with an extra 1; e.g. 16 = 10000, 17 = 10001, 18 = 10010, etc.

For an example of defining a multicolored pattern, consider a situation in which we have three bit planes and we want a two line (pixel) pattern. The first line is to be drawn entirely with palette 0, while the second line will consist of alternating short and long dashes with the short dashes being drawn in palette 2, the long dashes in palette 6 and the spaces between in palette 0. With our plain fill pattern, this process may have been defined as:

```
0000000000000000
0110111101101111
```

To translate this effect to our multicolor pattern, we must define three patterns- - - one for each bit plane, with the three vertical combinations, (if we imagine them "stacked vertically") defining the appropriate palettes.

So, we first should determine the binary values for the three palettes involved:

```
Palette 0 - 000
Palette 2 - 010
Palette 6 - 110
```

Recalling that our three "vertical" combinations must always be one of these values, we end up with:

Bit Plane 0:
```
0000000000000000 =&h0000
0000000000000000 =&h0000
```

Bit Plane 1:
```
0000000000000000 =&h0000
0110111101101111 =&h6f6f
```

Bit Plane 2:
```
0000000000000000 =&h0000
0000111100001111 =&h0f0f
```

Note that the sequence of the vertical combinations is important. In our example, palettes 0 and 2 are the same in either direction, but palette 6 (if we've got things back to front) could have ended up as 011 or palette 3! Bit plane 0 always represents binary digit furthest to the right .

Having defined our multicolored pattern, the next hurdle is to place that pattern in an

continued...

array. Well, we have six lines to place in the array, but we've already noted the array MUST be DIMensioned to a power of two. This part is easy! We just DIMension the array to the next power of two (DIM Pat%(8)) and leave the last portion unused. The data is then read into the array and the PATTERN statement is issued.

We now have another question to answer. How does Basic know this pattern is a two line, multicolored pattern and not an eight line fill pattern? The answer is that BASIC doesn't know. If we left things untouched, we would end up with an eight line fill pattern!

When Basic processes the PATTERN statement, it inserts a value representing the size of the pattern in the "RastPort" structure for the current window. This value is determined by the size of the specified array. If the size of the array is 2, 4, 8 or 16, etc, the size of the array is set to 1, 2, 3 or 4, etc, respectively. In our case, the size would have been set to three (for an eight line fill pattern), when we really want it set to one (for a two line fill pattern). But this clarification still doesn't indicate that we have a multicolored pattern. Multicolored is indicated by making the size a negative number. Thus, if the size is 3, we have an eight line fill pattern, while if it is -1, we have a two line, multicolor pattern.

We now must find the pattern size within the "RastPort" and change it to -1. The function WINDOW(8) provides the "RastPort" address, and since the pattern size is located at an offset of 29, it would seem reasonable to just issue the statement: POKE WINDOW(8)+29,-1. Unfortunately, this simple solution will not work! The pattern size is a one byte field and Basic always requires two bytes to store a negative number! We must get rid of those first eight bits. Actually, this work turns out to be very easy- - - we just ADD the negative number to 255 before POKEing it. Thus, POKE WINDOW(8),-1 AND 255 will set the correct pattern size.

One more task separates us from using the multicolored pattern. The background color must be set to 0 and the foreground color must be set to the highest available palette. This resetting may be accomplished with: COLOR WINDOW(6),0.

The multicolored pattern may now be used just like any other fill pattern. However, you should not specify any colors in LINE and/or PAINT statements, or issue another COLOR statement until you have finished with the pattern.

The Sample Programs

Two sample programs accompany this article. Enter and run them to see exactly what patterns can do for you! The first program shows a number of line and fill patterns. The most interesting point about this program is the use of a random pattern in one of the areas. Each time the program is run, the pattern changes!

The second sample program shows an example of a 16 line multicolored pattern which uses all eight colors available with the three bit planes which are used (the screen has a depth of 3). Once you have seen the result from this program, create a second version of it which loads the array with 48 lines of random numbers for a really striking effect! Use the method demonstrated in the first sample program and don't forget to add the 'RANDOMIZE TIMER' at the start of the program. (If you forget to Randomize, the pattern will not change each time you run the program).

As always, after entering the programs, save them before you try to "run."

As a final note, remember you are not restricted to a single pattern. Just load different patterns into different arrays during program initialization, and then use the PATTERN statement to switch between patterns! So, now that you know how to use patterns, dress up those graphs, bar charts or whatever and make a much greater visual impact!

Listing One

```
' Sample Program #1
' Line and fill patterns in AmigaBASIC
' Bryan D. Catley, April 1987
'
' Open a lo-res screen with a depth of 3, a full sized
' window with no gadgets, assign colors, clear the
' screen, and intialize random numbers.
'
SCREEN 2,320,200,3,1
WINDOW 2,,,0,2
PALETTE 0,0,0,0:Black=0
PALETTE 1,1,0,0:Red=1
PALETTE 2,1,1,0:Yellow=2
PALETTE 3,0,1,0:Green=3
PALETTE 4,0,1,1:Cyan=4
PALETTE 5,1,0,1:Magenta=5
PALETTE 6,0,0,1:Blue=6
PALETTE 7,1,1,1:White=7
COLOR ,Black:CLS
RANDOMIZE TIMER

' Draw circle with a wedge removed.
' Design is: 1111000011110000
'
COLOR Blue,White
PATTERN &HF0F0
CIRCLE (60,120),40,,-4.71,-3.14

' Now draw a triangle with a fill pattern
'
DIM Pat%(3)
FOR n=0 TO 3:READ x:Pat%(n)=x:NEXT
DATA &h03c0, &h03c0, &hffff, &hffff
PATTERN ,Pat%
ERASE Pat%
COLOR Yellow,Magenta
AREA(160,112):AREA STEP(-56,56)
AREA STEP(112,0):AREAFILL

' Time for a line pattern.
' Design is: 1010101010101010
'
COLOR Green,Blue
PATTERN &HAAAA
LINE (160,72)-(256,168)

' Time for circle with a fill pattern.
'
DIM Pat%(7)
FOR n=0 TO 7:READ x:Pat%(n)=x:NEXT
DATA &hf000, &h0f00, &h00f0, &h000f
DATA &h000f, &h00f0, &h0f00, &hf000
PATTERN ,Pat%
COLOR Blue,Yellow
CIRCLE (272,104),32
PAINT (272,104)

' Finally, we draw a rectangle filled with
' random pattern, and which is outlined with
' a dashed box.
'
FOR n=0 TO 7
  Pat%(n)=INT(RND*32000)
NEXT
PATTERN &HFFF0,Pat%
COLOR Magenta,White
LINE (8,8)-(304,64),,b
COLOR Yellow,Red
LINE (16,16)-(296,56),,bf

' Now wait for a mouse click to terminate.
'
COLOR Red,Black:LOCATE 24,14:PRINT"Click to Quit";
WHILE MOUSE(0)=0:WEND
```

```
' Reset pattern, (this is not really necessary,
' but let's be tidy), close the window and screen,
' and quit the program.
'
FOR n=0 TO 7:Pat%(n)=-1:NEXT
PATTERN -1,Pat%
WINDOW CLOSE 2
SCREEN CLOSE 2
END
```

Listing Two

```
' Sample Program #2
' Multicolor patterns in AmigaBASIC
' Bryan D. Catley, April 1987
'
' The sample pattern is a 16 line (pixel) cross hatch
' design using all 8 palettes.
'
' Open a lo-res screen with a depth of 3, a full sized
' window with no gadgets, assign colors, and clear the
' screen.
'
SCREEN 2,320,200,3,1
WINDOW 2,,,0,2
PALETTE 0,0,0,0:Black=0
PALETTE 1,1,0,0:Red=1
PALETTE 2,1,1,0:Yellow=2
PALETTE 3,0,1,0:Green=3
PALETTE 4,0,1,1:Cyan=4
PALETTE 5,1,0,1:Magenta=5
PALETTE 6,0,0,1:Blue=6
PALETTE 7,1,1,1:White=7
COLOR ,Black:CLS
' Dimension the array to hold the pattern.  With 3 bit
' planes we have to be able to store 48 lines (16x3),
' so the array is dimensioned to the next higher power
' of two.
'
DIM Pat%(63)
' Now place all 48 pattern lines in the array, ignoring
' the final 16 entries.
'
FOR n=0 TO 47:READ x:Pat%(n)=x:NEXT
' Pattern for Bit Plane 0:
'
DATA &H0055, &HFF55, &H0055, &HFF55:   ' Lines  1- 4
DATA &H0055, &HFF55, &H0055, &HFF55:   ' Lines  5- 8
DATA &H5500, &H55FF, &H5500, &H55FF:   ' Lines  9-12
DATA &H5500, &H55FF, &H5500, &H55FF:   ' Lines 13-16

' Pattern for Bit Plane 1:
'
DATA &H0033, &H0033, &HFF33, &HFF33:   ' Lines  1- 4
DATA &H0033, &H0033, &HFF33, &HFF33:   ' Lines  5- 8
DATA &H3300, &H3300, &H33FF, &H33FF:   ' Lines  9-12
DATA &H3300, &H3300, &H33FF, &H33FF:   ' Lines 13-16

' Pattern for Bit Plane 2:
'
DATA &H000F, &H000F, &H000F, &H000F:   ' Lines  1- 4
DATA &HFF0F, &HFF0F, &HFF0F, &HFF0F:   ' Lines  5- 8
DATA &H0F00, &H0F00, &H0F00, &H0F00:   ' Lines  9-12
DATA &H0FFF, &H0FFF, &H0FFF, &H0FFF:   ' Lines 13-16
' Tell Basic about the pattern.
'
PATTERN ,Pat%
' But Basic now thinks we have a 64 line standard fill
' pattern.  We must adjust this thinking to a 16 line
' multicolor pattern.  (See text of accompanying
' article for further explanations).
'
POKE WINDOW(8)+29,-4 AND 255
' Before we can use the pattern, we must set the fore-
' ground color to the highest palette available, and
' the background color to zero.
'
COLOR WINDOW(6),0
' Finally, we can use our multicolored pattern...
' Let's try a triangle first...
'
AREA(64,72)
AREA STEP(-56,104)
AREA STEP(112,0)
AREAFILL
' Now a circle...
'
CIRCLE (153,72),64
PAINT (153,72)
' Finally, a rectangle...
'
LINE(240,8)-STEP(64,168),,bf
' Now wait for a mouse click to terminate.
'
COLOR Red,Black:LOCATE 24,14:PRINT"Click to Quit";
WHILE MOUSE(0)=0:WEND
' Now close the window screen, (this also resets
' the pattern to solid); and quit the program.
'
WINDOW CLOSE 2
SCREEN CLOSE 2
END
```

•AC•

Amiga Developers

This space could have been yours at a rate you wouldn't believe. Amazing Computing™ is the Amiga users' magazine, dedicated to helping the Amiga user get the most out of his machine. Amazing Computing™ is now sold at over 1200 locations worldwide and is read by users who want to do more with their Amigas. These Amiga users are searching for an application program or hardware item you have developed.

If you are an Amiga Developer who is trying to get the most cost effective advertising for a new product (without running a budget like the U.S. Government) then give us a call. If money is no object, we're certain other magazines will also be glad to help!

Contact:

John Fastino
PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722
(617) 678-4200

Amazing Computing™: Your Resource to the Commodore Amiga™

Programming with SoundScape:
Manipulating samples in the Sampler module

by Todor Fay
Author of SoundScape

Introduction

In Amazing Computing V2.5, I covered the basics of programming with SoundScape--How to get the simplest module up and running and receiving and sending MIDI music events. In order to best understand what follows, it would be a good idea to read that article if you haven't.

Some of the SoundScape modules allow other modules and programs access to important data structures. This process is done through the mechanism of State Structures and Edit Routines discussed last month. In this article, I'll talk about one particular module, the Sampler. I'll discuss how it works, examine its State Structure and write an example program that manipulates it by letting the user put cross faded loops into selected samples.

The Sampler Module

The Sampler is the sound generating module in SoundScape. It receives MIDI note events and converts them into audible pitched sounds, using the Amiga's audio device. As a MIDI device, it responds appropriately to note velocity information and MIDI pitch bend events. Since there are sixteen MIDI channels, the Sampler supports all sixteen. Each channel can be a different musical instrument and each can have up to ten octaves of samples.

These samples can be loaded and saved from files on a channel-by-channel basis or as a group. Both IFF and SoundScape file formats are supported. There is an editor that allows editing of the individual samples and the setting of loop points graphically. There is also the Mimetics sampler hardware that can be used to create these samples.

How the Sampler processes MIDI events

When the Sampler receives a MIDI NOTEON event, it checks the channel number (carried in the status byte of the MIDI event.). If there isn't any instrument loaded on that channel, the MIDI event is discarded. Otherwise, the sampler takes the note value information in the event and breaks it into two parts: octave and note in that octave. It then checks if there is a sample loaded for that octave. If so, the sampler allocates one of the four Amiga voices.

If there is no voice free, it must steal one of the currently playing voices. It takes the one that is farthest along in its envelope. This voice is not neccessarily the one that has been playing the longest. If one note has been playing for several seconds and is still in its sustain section, while a much shorter staccato note is already being released, it makes obvious sense to steal the latter. A message is sent to the Amiga's audio device, telling it to stop playing the stolen voice (CMD_FLUSH).

A sample is played by dumping the majority of the sample out, then holding on a looped section. The looped section is neccessary because it is far cheaper to play the same part over and over again, rather than having very long samples to accomodate the longest possible playing of the sound.

Looping is handled by sending two write commands (CMD_WRITE) to the audio device. The first has the sample from the very beginning to the end of the loop. It asks the audio device to play this just once. The second command controls just the section of the sample between the two loop points. The audio device is instructed to loop on this section forever.

Envelopes are handled by a 60hz vertical interrupt routine. Each sample has an envelope that is broken into four slope segments and a sustain portion. Each slope is defined by a rate and a level to go to. The interrupt routine checks which slope segment this voice currently is in and simply adds the rate to the current volume to determine the new volume, which it then sets. When this volume reaches the level for the current slope segment, the interrupt routine places the envelope in the next slope. When the sustain portion of the envelope is reached, the volume is held at the sustain level and no volume changes are made.

When a NOTEOFF MIDI event arrives, the envelope for this note is advanced into its final decay slope segment. The interrupt routine resumes changing its volume until it finally decays to nothing. Then, the interrupt routine sends a CMD_FLUSH message to the audio device and this voice is free.

MIDI note velocity is handled by simply multiplying the volume by it. So, note emphasis directly translates to loudness.

Data Structures

For each sample, there is a structure that keeps all the data that pertains to it:

```
struct SoundData {
        char *data;
        UWORD loopstart;
        UWORD loopend;
        UWORD length;
        UWORD start;
};
```

The first field, 'data', is a pointer to the actual sample somewhere down in chip memory. This sample can be any length up to 65k, and the field 'length' stores that. 'Loopstart' is an offset into the sample for the looped part of the sample to start. 'Loopend' is an offset into the sample for the end of the loop. 'Start' is an offset into the sample that indicates where the sample should actually start playing. This comes in handy when someone is editing a sample and is trying to decided where in it playback should start.

Each channel has ten of these SoundData structures for the ten samples. In addition, it needs to keep track of the envelope for this sound, a transposition value, velocity and pitchbend sensitivity, and more. So, for each channel, there is the ChannelData structure:

```
struct ChannelData {
        int rate1, rate2, rate3, rate4;
        int level1, level2, level3;
        long pad(8);
        unsigned short sensitivity;
        unsigned short start;
        unsigned short loopstart;
        unsigned short loopend;
        unsigned short pbendamount;
        unsigned short pbendrange;
```

continued...

```
        short tune;
        struct SoundData sounddata[10];
        unsigned short pbend[129];
        char name[20];
        char transpose;
        char status;
        char type;
};
```

Many of these fields are for internal use, and explanations would be far more lengthy and boring than instructive. However, there are a few that could be of some interest. The rate and level fields correspond directly with the envelope sliders in the Sampler window. By changing these, you change the envelope of the sounds played on this channel. The transpose field is simply a number to add to incoming note values, thereby shifting them in key and, for larger transpositions, also in octave.

The field of most interest is the array of SoundData structures. This is where all samples for this channel reside. With access to this, you can manipulate the samples. Access to these structures is made possible by the SoundScape mechanism of state structures and edit routines. So, we get to one last data structure, the State Structure for the Sampler.

A little background for those who haven't read the V2.5 article: Each SoundScape module can make some of its variables available to the outside world. It does this by defining a structure with these variables in it. This is called a 'State Structure' because it holds the state of that module at any given time. Changing the structure sets the module to another state. This is very useful for environment loads and saves. The environment save command simply asks all the modules in the Patch Panel for a copy of their state structures; it then stores all of them in one file. The load command simply reads that file, and returns all the modules to their state structures. Because each state structure always starts with a length field, the load and save commands can work with state structures of all shapes and sizes without having to know what's in them.

```
struct SamplerState {
        long length;
        char namesfile[40];
        struct ChannelData *channeldata[16];
        struct EnvelopeInfo *envelopeinfo;
        struct MsgPort *soundport;
        long addchannel;
};
```

'Namesfile' is the name of a file that has the names of the instruments on all sixteen channels. To load a set of instruments, this file is read, then the instruments it defines. In this way, instruments can be kept in seperate files, while there is also a method for loading a set of them (of particular interest to environment loads and saves.)

The last two fields, 'envelopeinfo' and 'soundport' are of little use and confusing at best.

Of great interest is the array of sixteen ChannelData structures. These are pointers and you can only read them, the Sampler module will not let you change them. Because they take up a lot of memory, only the channels being used have their ChannelData structures allocated. However, if the ChannelData structure for the channel you require is not allocated, you can signal the Sampler that you would like it allocated by setting the field 'addchannel' to the channel number and executing an EditMidiPort SETSTATE call (which instructs the Sampler to use this state structure data. Then, you call EditMidiPort with the GETSTATE command and the state structure should now have the ChannelData structure you wanted allocated.

Before you tear your hair out with the prospect of messing with all these EditMidiPort calls, a faster, less messy way comes to you rescue.

There is a file you can link in with your program called "extras.o" that has several routines that do all the messy work of communicating with the Sampler through EditMidiPort calls.

Routines of interest are:

GetChannelData(channel)

This routine returns a pointer to the ChannelData structure specified by the given channel number (0 - 15). If it doesn't exist yet, it is allocated.

GetSoundData(channel,octave,sounddata)

This routine is very useful. You specify which octave and channel you would like to access the sample of. You also give it a SoundData structure. GetSoundData goes to the channel (by calling GetChannelData) and copies the SoundData structure for the selected octave into your SoundData structure. It also makes a copy of the sample and gives that to you.

There is a good reason for giving you copies and not the originals. If you receive the original, it is entirely possible for another module, or maybe just the user playing with the edit window, to make changes that conflict with what you are doing. For example, one program could return the sample to the system free list, while the other is editing it. By giving you only a copy, this kind of problem is avoided.

Because GetSoundData has allocated memory for the sample copy, you do have the responsibility of freeing it when you are done:

FreeMem(sounddata->data,sounddata->length);
SetSoundData(channel,octave,sounddata)

Use this to give a sample to the Sampler.

This takes the SoundData structure you provide and copies it into the SoundData structure for the channel and octave you indicate. Once again, everything is copied, including the sample itself. So, you still need to free your copy of the sample when you are done with it.

Finally, after all this talking, we have a reasonable understanding of how the Sampler works, and we have the tools to monkey with it, in particular with samples. So let's write a simple but very useful module.

Cross Fade Loop Module

It's very hard to make loops that sound good. Even with loop boundaries landing on zero crossings (removing the "click" produced by the sharp transient if they aren't), most loops have an audible bounce to them. This is because the nature of the sound is rarely the same at the beginning and end of a loop. The timbre, amplitude, sometimes even pitch, usually don't match. The loop ends up sounding something like "wadoingdoingdoingdoingdoing". One solution is to have the loop be only one cycle in length. Now, there is no chance for the sound to change. But, it tends to sound very stale and organ-like since it isn't changing.

Cross fade looping to the rescue! If you can make the end of the sample match the beginning of the sample, you can get very nice loops. A software generated cross fade loop does just this by gradually mixing the portion of the sample that precedes the loop into the portion at the end of the loop. By the end of the loop, it takes over entirely, so it is identical to the sample entering the loop. This method is called "cross fade looping" because it fades from one sound to the other.

Our cross fade loop module is very simple. There is no need to send or receive MIDI events. There is no need to support environment loads and saves. Just provide a routine that is invoked when the user clicks on the icon for this module. This routine puts up a window with gadgets for channel and octave of the sample to crossfade. There is also a boolean gadget that commands the module to crossfade the specified sample and an undo gadget to return the sample to its previous condition.

The crossfade command uses a GetSoundData call to copy the sample specified by the octave and channel gadgets into its buffer. It also makes a copy into the undo buffer (another SoundData structure.). The crossfade is computed and returned to the Sampler with a SetSoundData call.

The undo command simply makes a SetSoundData command with the undo buffer and then clears that buffer.

In order to make it easy for the user to test the results, the Console Keyboard is automatically hooked up to the Sampler when the cross fade module is first opened. This is done using two SoundScape calls:

FindMidiPort(portname)

Given the string name of a SoundScape module, this returns the actual port id of it. This is neccessary because all SoundScape calls use these numbers rather than names for reasons of speed. There is an opposite function, MidiPortName(id), which returns the pointer to a port's name, given its id. This is useful for finding what modules are out there.

OpenLink(sourceid,destinationid)

This routine adds a connection in the Patch Panel connecting the two modules designated by their id numbers. 'Sourceid' sends to 'destinationid'. Both modules are activated by this command. This is the same operation as when the user clicks on two icons in the Patch Panel, connecting them. To remove a connection, the command CloseLink(sourceid,destinationid) does the job.

So, in the open routine for the cross fade module, FindMidiPort is called twice to find the ids of "sampler" and "console keyboard". These ids are then used by OpenLink to connect them.

(see the listing for crossfade.c)

Here's the with file to link this with:

```
FROM *
Lib:Astartup.obj,*
sslink.o,*
crossfade.o,*
extras.o,*
morelib.o
TO
CrossFade
LIBRARY *
Lib:amiga.lib,*
Lib:lc.lib
```

To run this, first get SoundScape running. Run the cross fade program from the CLI (unless you've made WorkBench icon for it.) The cross fade icon will appear on the left side of the Patch Panel. Click on it twice. A small window with the four gadgets appears. Octave is preset to 5 and channel to 1. Notice that the Console Keyboard opens and connects to the Sampler. Open the Sampler. Open the first sound (channel 1). Load a sample or go directly to the Sample Capture & Edit window and make a sample in octave five by either drawing it or sampling it (if you have the sampler hardware.) Make your loops and listen to them. Kalamazoinkzoinkzoink...

Go back to the cross fade window and hit the cross fade button. Nothing happens visually because the Sampler window doesn't know you monkeyed with its sample. Get it to re-disply by clicking on the sizing icon on its lower right hand corner. Voila! The region in the loop section has changed. Listen to it. Kalamazooooooooooo... Hit the Undo button. Listen again. Kalamazoinkzoinkzoink...

That's it! Have fun cleaning up your library of samples.

continued...

Where do we go from here?

Now that you can grab samples, do weird and beautiful things to them, then return them to the Sampler, there are lots of ideas to experiment with:

• A module with several filters for processing samples.

• A module that mixes two or more samples.

• A module that simply creates samples, given an intelligent, or completely wacky algorithm.

Feel free to take this program and mutate it in one hundred one wondrous ways.

The code for this program, the neccessary files to link with and the resulting object file can be obtained from PeopleLink and a future Amicus disk. Files for both Aztec and Lattice C are provided.

Listing One

```
/*CROSSFADE.CCross fade loop generator for the
SoundScape sampler.

(c) 1987 Todor Fay
*/

#include "exec/exec.h"
#include "exec/types.h"
#include "intuition/intuition.h"
#include "soundscape.h"
#include "sampler.h"

/*Cross Fade Icon for Patch Panel: */

UWORD crossfadedata[] = {/* 36 x 11 */
0,0,0,
16383,65535,49152,
511,65528,0,
15887,65287,49152,
16368,24831,49152,
16383,40959,49152,
16368,24831,49152,
15887,65287,49152,
511,65528,0,
16383,65535,49152,
0,0,0,
65535,65535,61440,
49152,4,12288,
49152,57544,12288,
49153,61700,28672,
55792,24812,61440,
65535,40959,61440,
55792,24812,61440,
49153,61444,28672,
49152,57544,12288,
49152,4,12288,
65535,65535,61440,
};

struct Image crossfadeimage =
{ 0,0,36,11,2,crossfadedata,3,0,0 };

/*Edit window.  This has the gadgets for selecting
sampler channel and octave, and Doing and Undoing
the cross fade.  These intuition data structures
were generated with Power Windows.
*/

USHORT BorderVectors1[] = {0,0,107,0,107,10,0,10,0,0};
struct Border Border1 = {
-2,-1,
3,0,JAM1,
5,
BorderVectors1,
NULL
};

struct IntuiText IText1 = {
1,0,JAM2,
33,1,
NULL,
"Undo",
NULL
};

struct Gadget UndoGadget = {
NULL,
23,55,
104,9,
GADGHCOMP,
RELVERIFY,
BOOLGADGET,
(APTR)&Border1,
NULL,
&IText1,
0,
NULL,
8,
NULL
};

UBYTE SIBuffer7[4] =
"5";
struct StringInfo GadgetSI7 = {
SIBuffer7,
NULL,
0,
4,
0,
0,0,0,0,0,
0,
5,
NULL
};

USHORT BorderVectors2[] = {0,0,35,0,35,9,0,9,0,0};
struct Border Border2 = {
-2,-1,
1,0,JAM1,
5,
BorderVectors2,
NULL
};

struct IntuiText IText2 = {
2,0,JAM2,
-60,0,
NULL,
"Octave",
NULL
};

struct Gadget OctaveGadget = {
&UndoGadget,
95,28,
32,8,
GADGHCOMP,
LONGINT+RELVERIFY+STRINGCENTER,
STRGADGET,
(APTR)&Border2,
NULL,
&IText2,
0,
(APTR)&GadgetSI7,
7,
NULL
};

UBYTE SIBuffer6[4] =
"1";
struct StringInfo GadgetSI6 = {
SIBuffer6,
```

```
NULL,
0,
4,
0,
0,0,0,0,0,
0,
1,
NULL
};

USHORT BorderVectors3[] = {0,0,35,0,35,9,0,9,0,0};
struct Border Border3 = {
-2,-1,
1,0,JAM1,
5,
BorderVectors3,
NULL
};

struct IntuiText IText3 = {
2,0,JAM2,
-67,0,
NULL,
"Channel",
NULL
};

struct Gadget ChannelGadget = {
&OctaveGadget,
95,15,
32,8,
GADGHCOMP,
LONGINT+RELVERIFY+STRINGCENTER,
STRGADGET,
(APTR)&Border3,
NULL,
&IText3,
0,
(APTR)&GadgetSI6,
6,
NULL
};

USHORT BorderVectors4[] = {0,0,107,0,107,10,0,10,0,0};
struct Border Border4 = {
-2,-1,
3,0,JAM1,
5,
BorderVectors4,
NULL
};

struct IntuiText IText4 = {
1,0,JAM2,
12,1,
NULL,
"Cross Fade",
NULL
};

struct Gadget CrossFadeGadget = {
&ChannelGadget,
23,41,
104,9,
GADGHCOMP,
RELVERIFY,
BOOLGADGET,
(APTR)&Border4,
NULL,
&IText4,
0,
NULL,
5,
NULL
};

/* Gadget list */
```



```
struct NewWindow NewWindowStructure = {
0,0,
152,72,
0,1,
GADGETUP+CLOSEWINDOW+RAWKEY,
WINDOWSIZING+WINDOWDRAG+WINDOWDEPTH+WINDOWCLOSE
+ACTIVATE,
&CrossFadeGadget,
NULL,
" X Fade ",
NULL,
NULL,
30,20,
152,72,
WBENCHSCREEN
};

unsigned short thisport;
opencode(direction)

/*There is nothing to be initialized.  However,
it would be nice to connect the console
keyboard to the sampler so the user can test
the cross fade loops, so make an OpenLink call.
FindMidiPort returns the port id, given the
name of the port.  Actually, these two ports
have ids that are hard coded into SoundScape.
This is an example what you should do with
modules that are independently loaded into
the system.
*/

unsigned char direction;

{
    OpenLink(FindMidiPort("console keyboard"),
```

continued...

```
FindMidiPort("sampler"));
    return(1);
}

closecode(direction)

unsigned char direction;

{
    return(1);
}

clearsounddata(sounddata)

/*This routine frees up the sample buffer belonging
to the SoundData structure 'sounddata'.
*/

struct SoundData *sounddata;

{
    if (sounddata->data)
FreeMem(sounddata->data,sounddata->length);
    sounddata->data = 0;
    sounddata->length = 0;
    sounddata->loopstart = 0;
    sounddata->loopend = 0;
    sounddata->start = 0;
}

crossfade(sounddata)

/*This, the cross fade routine, operates on a
SoundData structure.

The cross fade is done on the region of the
sample between the two loop points.  The
sample prior to the loop start is gradually
mixed in until it is in full force by the
end of the loop.

This is very straight forward, the only problem
encountered is a sample with a loop that
is longer than the part leading into the
loop.  This is solved by having the cross fade
section be shorter - the length of the lead
in section.
*/
struct SoundData *sounddata;
{
    char *sample;/* Points to sample */
    unsigned short loop;/* Length of loop. */
    unsigned short crosslength;/* Length of cross fade. */
    unsigned short start;/* Start point. */
    long result;/* Intermediate result. */
    unsigned short a,b;/* Indexes into sample. */
    unsigned short j;/* Loop index. */

    loop = sounddata->loopend - sounddata->loopstart;
    sample = &sounddata->data[sounddata->start];
    start = sounddata->loopstart - sounddata->start;
/*If the loop is greater than the start segment, set
the crossfade to cover the last part of the loop,
equal to the starting segment.
*/

    if (loop > start) {
crosslength = start;
a = 0;
b = loop;
    }
/*Otherwise, set the crossfade to cover the entire
loop.
*/

    else {
crosslength = loop;
a = start - loop;
b = start;
    }
    for (j = 0; j < crosslength; j++ ) {
result = (j * sample[a]);
result += (crosslength - j) * sample[b];
result = result / crosslength;
sample[b] = result;
a++;
b++;
    }
}
void useredit()

/*This routine is called whenever the user clicks
twice on the cross fade icon.  A window is opened
and four gadgets are presented.  Two string
gadgets allow the user to select the octave and
channel of the sample they'd like to work on.
Two boolean gadgets are used to command the
cross fade and undo operations.

There are two SoundData structures.  OldSound
is used to keep an old copy of the sample
around for undoing.  CFSound is used to hold
the sample that gets cross faded.

RAWKEY events are passed on to the console
keyboard with the OutConsole command.
This is so the user can test the changed sound
without having to click on another window to
enable the Console Keyboard.
*/

{
    struct IntuiMessage *message;
    struct SoundData *oldsound, *cfsound;
```

```
    short code, class;
    struct Gadget *gadget;
    struct Window *window;
    short octave = 5;
    short channel = 0;
    oldsound = (struct SoundData *)
AllocMem(sizeof(*oldsound),MEMF_CLEAR);
    if (!oldsound) return;
    cfsound = (struct SoundData *)
AllocMem(sizeof(*cfsound),MEMF_CLEAR);
    if (!cfsound) {
FreeMem(oldsound,sizeof(*oldsound));
return;
    }
    window = (struct Window *)
OpenWindow(&NewWindowStructure);
    while (window) {
while (!(message = (struct IntuiMessage *)
  GetMsg(window->UserPort)))
    WaitPort(window->UserPort);
class = message->Class;
code = message->Code;
gadget = (struct Gadget *) message->IAddress;
ReplyMsg(message);
if (class == CLOSEWINDOW) break;
if (class == RAWKEY) OutConsole(code);
if (class == GADGETUP) {
    switch (gadget->GadgetID) {
case 5 :

/*The crossfade gadget.  Get the sample, using
GetSoundData, and put it in cfsound.  Clear
the undo buffer (oldsound) and get the same
sample loaded into it.  Call crossfade to alter
cfsound, then install it in the sampler with
a call to SetSoundData.
Then, clear out cfsound.
*/
    if (!(GetSoundData(channel,
      octave,cfsound))) {
clearsounddata(oldsound);
if (GetSoundData(channel,
  octave,oldsound)) {
    oldsound->data = 0;
}
crossfade(cfsound);
SetSoundData(channel,octave,cfsound);
clearsounddata(cfsound);
    }
    break;
case 6 :

/*This gadget simply sets the channel of the sampler
that we are looking at.  Also, clear the undo buffer.
*/
    channel = (GadgetSI6.LongInt - 1) & 0x0F;
    clearsounddata(oldsound);
    break;
case 7 :

/*This gadget sets the octave of the sample we are
looking at.  Clear the undo buffer.
*/
    octave = GadgetSI7.LongInt;
    if (octave < 0) octave = 0;
    if (octave > 9) octave = 9;
    clearsounddata(oldsound);
    break;
case 8 :

/*This is the undo command.  Simply give the undo
buffer to the sampler, then clear the undo buffer.
*/
    if (oldsound->data) {
SetSoundData(channel,octave,oldsound);
clearsounddata(oldsound);
    }
    break;
    }
}
```



```
    }
    CloseWindow(window);
    clearsounddata(oldsound);
    clearsounddata(cfsound);
    FreeMem(oldsound,sizeof(*oldsound));
    FreeMem(cfsound,sizeof(*cfsound));
}
editcode(direction,command,buffer)

/*This is the edit routine provided to SoundScape
in the AddMidiPort call.
There is nothing in the way of data structures
that we would like any other modules to access,
nor is there anything to save or load with
environment loads and save.  So, the GETSTATE,
SETSTATE, SAVESTATE, and LOADSTATE commands are
all ignored and the sample's length is set to
0.

However, USEREDIT indicates the user has clicked
twice on the cross fade icon and would like to
work with it. In that case, call the routine
'useredit'
*/

char direction, command;
long *buffer;
{
    static char not_editing = 1;
    *buffer = 0;
    switch (command) {
case USEREDIT :
    if (not_editing) {
not_editing = 0;
useredit();
not_editing = 1;
    }
    break;
    }
}

/*The main program */
long SoundScapeBase;
long IntuitionBase;
main() {
    SoundScapeBase = OpenLibrary("soundscape.library",0);
    if (SoundScapeBase) {
IntuitionBase = OpenLibrary("intuition.library",0);
CloseLibrary(SoundScapeBase);
if (Version() > 1) {
    thisport = AddMidiPort(opencode,closecode,
editcode,0,&crossfadeimage,0,-1,"cross fade");
    SetTaskPri(FindTask(0),-20);
    while (MidiPort(thisport)) Delay(100);
}
CloseLibrary(IntuitionBase);
    }
}
```

•AC•

Explore the Potential of Amiga!

NEW YORK	LOS ANGELES	CHICAGO
October 10-12, 1987	January 22-24, 1988	July 22-24, 1988

Featuring

Keynote Sessions

Jay Miner, the Father of the Amiga™, will open the New York **AmiEXPO.** ***R. J. Mical,*** the Designer of Intuition, will provide insights into software development.

Exhibition Hall

A sampling of exhibitors:

Activision, Inc.
Amazing Computing
Amigo Business Computers
ASDG, Inc.
Brown-Wagh Publishing
Byte by Byte
Central Coast Software
Computer Systems Associates
Creative Microsystems, Inc.
Firebird Licensees, Inc.
Lattice, Inc.
Liquid Light, Inc.
Manx Software Systems
Microillusions
NewTek, Inc.
New Horizons Software
PiM Publications, Inc.
Octree Software
Word Perfect Corporation

Amiga™ is a registered trademark of Commodore-Amiga, Inc.

Development Forums

Intensive working sessions with the leading Amiga developers, such as *NewTek,* detailing specific products.

User Seminars

- The Architect's Amiga
- Art Direction and the Amiga
- The Amiga in Video Production
- Amiga MIDI: Lights, Sound, Action!
- The Ultimate Game Machine, Amiga
- Amiga's Desktop Color Publishing
- "Vax in a Box" - Amiga Engineering

* AND MORE *

For information call ***800-32-AMIGA*** *(in New York call 212-867-4663) or complete the form and return it to:*

AmiEXPO Headquarters

YES! Send me more info on **AmiEXPO!** AC6

Name _____
Company _____
Address _____
_____
_____
Telephone _____

Return form to:
AmiEXPO Headquarters
301 East 43rd Street, Suite 301
New York, New York 10017

The AMICUS Network

Courting C-64 owners, Animation formats

by John Foust

Commodore wants every Commodore 64 owner to buy an Amiga 500. More than six million 64s have been sold since its introduction in the early 1980s, making it one of the most popular computers of all time. Commodore hopes to essentially double the world-wide installed base of Amiga-compatible machines by years' end.

How will this be accomplished? In mid-summer, Commodore will woo 225,000 64 users with a special mailing detailing a series of new promotions. A special software offer is at the heart of the mailing. Keynote Amiga software is bundled together with an Amiga 500 at a special low price.

How low? For $99, you can get Deluxe Paint II, Aegis Animator, Marble Madness, Textcraft Plus, Pagesetter and an Epyx joystick. A second software bundle costs $199 and contains Word Perfect, Superbase Personal, Pagesetter with LaserScript, a version of Maxiplan for the Amiga 500, CLI Mate, Diga, and Deluxe Video. The retail value of this software tops $1500. The promotion runs from August 15 to October 31. You need to buy an Amiga 500 from a participating dealer to qualify for this offer.

Here in the United States, Kellogg's cereals will carry a promotion that has Amiga 500s as prizes. In Germany, two hundred Amiga 500s will be given away as part of a game card promotion in cooperation with McDonald's restaurants. The Amiga 500 will be advertised on television and in the point-of-purchase displays in the restaurant.

Animation formats

At this writing, Aegis has shipped VideoScape 3D and Byte-by-Byte has shipped Sculpt 3D, adding to the list of advanced video products for the Amiga. Animator:Apprentice from Hash Enterprises and Forms in Flight from Micro Magic have been shipping for months. Electronic Arts recently acquired the consumer version of the long-awaited Caligari animation program, so it may be out soon.

One concern of many users is that the new video programs do not share a common file format for objects. If objects were portable between programs, then time and effort would not be wasted if you wanted to use the objects created with one program in another.

In some object modeling or animation programs, you design three-dimensional objects as a collection of connected polygons. If you made a round object, it would still be composed of many small flat polygons. The polygons are given characteristics such as color and texture. When the descriptions of these objects are saved to disk, they won't have the same interchangability between animation programs as compared to Amiga paint programs that use the IFF standard, for example.

The user interface of the object editor varies widely in each program. Sculpt and Forms in Flight have interactive, graphic interfaces that display the object being created on the screen at all times. (Strictly speaking, Sculpt is not an animation program. From an object creation standpoint, it is similar to other animation programs.)

VideoScape's object editor is almost non-existant. The program includes utilities for creating simple objects such as boxes, cylinders and surfaces of revolution such as a wine glass or aircraft body. It is text-oriented, not graphic. Another simple graphic object editor is included with VideoScape, based on Colin French's public domain ROT program. It knows nothing of primitive objects, but only points, lines and polygons.

Note that a standard exists for the resulting animations, though. The ANIM format only defines the form of the final product, the animation file itself. An ANIM file can be considered as a complex IFF picture. If you try to load an ANIM file into Deluxe Paint, the first frame of animation will be loaded as a picture. Without a more sophisticated program, the other frames are not accessible. Aegis Development has acknowledged the existence of an "EditANIM" program that will allow changes to ANIM files, and other developers are working on similar programs.

While ANIM is defined and generally accepted, there may be no such standard for object format in the near future. It may take another generation of Amiga video products to implant a standard file format for objects. So far, the closest thing to a common format is an ASCII file that contains lists of points to be connected as polygons. VideoScape works solely with ASCII files. Sculpt can load an ASCII script file as well as load and save its own proprietary binary format that resembles an IFF type.

At this time, animation programs are sufficiently different from a programming standpoint that sharing objects is difficult, and impossible without specialized conversion software.

For example, compare the objects in VideoScape and Sculpt 3D. In VideoScape, every object is built up from arbitrary plane polygons. Each flat polygon in a VideoScape object can have any number of points. Sculpt builds up objects from triangular polygon faces. Sculpt objects could be converted to VideoScape without much trouble because its triangle objects are a subset of VideoScape objects. Moving objects in the other direction (from VideoScape to Sculpt) will be difficult until there is a conversion program that changes the complex polygon faces to simple triangles, and then outputs a file in Sculpt form.

To complicate matters, Forms in Flight allows even more flexibility in objects. Polygons can have many points, as well as being twisted and apparently non-contiguous, as long as the points are still in a single plane. Imagine twisting a dollar bill to make it look like a bow tie, then flattening it with the twist in place. This type of polygonal object is incompatible with VideoScape and Sculpt.

Some programs do not use polygons at all. Animator:Apprentice objects are made of slices, not polygons. The edges of the slices are colored. When the slices are stacked, they look like the object you want.

continued...

I have also seen ray-tracing modeling programs that use true spheres and surfaces of revolution to compose objects. For instance, the original Juggler demo has objects made up of only spheres. Again, all these object description methods are incompatible without complex conversion programs.

There is another dimension of incompatibility. These programs will disagree on environmental settings such as lighting. Some animation programs allow the user to place light sources to illuminate a scene. Some programs allow an indefinite number of light sources, some allow only one. Sculpt, Forms in Flight and Animator:Apprentice allow objects of any color, for 4096 choices, while VideoScape is limited to low resolution and a reduced set of colors, about fifty, using dithered shades of a fixed palette.

Other considerations include the settings for reflectance, texture, smoothing. Reflectance describes how much light is reflected from an object, and in what way. A mirrored object reflects light very well and in a coherent fashion. A dull object reflects light in a different way. There are transmissive characteristics for objects like colored glass. In the best of programs, glass can even refract light, so you could create an object that acts like a lens - say a crystal ball - and then view the scene through the lens.

Does this mean object compatiblility is a lost cause? I don't think so. Some programs will always use novel methods of describing objects. Other Amiga developers could adopt an object file format standard from another software community, such as CAD file formats such as IGES from minicomputers or mainframes. Tomorrow's animation and modeling programs would change around the standard. The new file format would enforce a programming methodology, so programs would be more likely to carry similar features for coloring, reflectance and lighting.

SIGGRAPH

Next month, Amazing Computing will have a report from SIGGRAPH, the Association of Computing Machinery's special interest group on graphics. It is being held in Anaheim, California. Commodore will have a booth at the show. This convention attracts the best and the brightest of the computer graphics industry.

I also hope to visit the set of Max Headroom at the Lorimar TelePictures studio in Culver City, California. Former Commodore employee Jeff Bruette, who worked on the Amazing Stories episode described in this summer's video issue, is now the technical consultant for the Max Headroom television series. The Amiga is being used for graphics and special effects.

•AC•

Index of Advertisers

The AMICUS & Fred Fish

Public Domain Software Library

This software is collected from user groups and electronic bulletin boards around the nation. Each Amicus disk is nearly full, and is fully accessible from the Workbench. If source code is provided for any program, then the executable version is also present. This means that you don't need the C compiler to run these programs. An exception is granted for those programs only of use to people who own a C compiler.

The Fred Fish disk are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Note: Each description line below may include something like 'S-O-E-D', which stands for 'source, object file, executable and documentation'. Any combination of these letters indicates what forms of the program are present. Basic programs are presented entirely in source code format.

AMICUS Disk 1

ABasic programs: Graphics

3DSolids	3d solids modeling prog. w/sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program, w/ hidden line removal
JPad	simple paint program
Optical	draw several optical illusions
PaintBox	simple paint program
Shuttle	draws the Shuttle in 3d wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws color spirals
ThreeDee	3d function plots
Topography	artificial topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes

ABasic programs: Tools

AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasic programs from a menu
MoreColors	way to get more colors on the screen at once, using aliasing
shapes	simple color shape designer SpeakIt speech and narrator demo

ABasic programs: Games

BrickOut	classic computer brick wall game
Othello	also known as 'go'
Saucer	simple shoot-em-up game
Spelling	simple talking spelling game
ToyBox	selectable graphics demo

ABasic programs: Sounds

Entertainer	plays that tune
HAL9000	pretends it's a real computer
Police	simple police siren sound
SugarPlum	plays "The Dance of the Sugarplum Fairies"

C programs:

ATerm	simple terminal program, S-E
cc	aid to compiling with Lattice C
decvnt	opposite of CONVERT for cross developers
Dotty	source code to the 'dotty' window demo
echox	unix-style filename expension, partial S,O-D
fasterfp	explains use of fast-floating point math
FixDate	fixes future dates on all files on a disk, S-E
freedraw	simple Workbench drawing prog.,S-E
GfxMem	graphic memory usage indicator, S-E
Grep	searches for a given string in a file with
ham	shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfers between an IBM and an Amiga
Mandel	Mandelbrot set program, S-E
moire	patterned graphic demo, S-E
objfix	makes Lattice C object file symbols visible to Wack, S-E
quick	quick sort strings routine
raw	example sample window I/O
setiace	turns on interlace mode, S-E
sparks	qix-type graphic demo, S-E

Other executable programs:

SpeechToy	speech demonstration
WhichFont	displays all available fonts

Texts:

68020	describes 68020 speedup board from CSA
Aliases	explains uses of the ASSIGN command
Bugs	known bug list in Lattice C 3.02
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
Filenames	AmigaDOS filename wildcard conventions
HalfBright	explains rare graphics chips that can do more colors
ModemPins	description of the serial port pinout
RAMdisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of Instrument demo sound file format
Speed	refutation of Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

AMICUS Disk 2

C programs:

alib	AmigaDOS object library manager , S-E
ar	text file archive program, S-E
fixobj	auto-chops executable files
shell	simple CLI shell, S-E
sq, usq	file compression programs, S-E
YachtC	a familiar game, S-E
Make	a simple 'make' programming utility, S-E
Emacs	an early version of the Amiga text editor, S-E-D

Assembler programs:

bsearch.asm	binary search code
qsort.asm	Unix compatible qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function, O-D

(This disk formerly had IFF specification files and examples. Since this spec is constantly updated, the IFF spec files have been moved to their own disk in the AMICUS collection.)

John Draper Amiga Tutorials:

Animate	describes animation algorithms
Gadgets	tutorial on gadgets
Menus	learn about Intuition menus

AMICUS Disk 3

C programs:

Xref	a C cross-reference gen., S-E
6bitcolor	extra-half-bright chip gfx demo, S-E
Chop	truncate (chop) files down to size, S-E
Cleanup	removes strange characters from text files
CR2LF	converts carriage returns to line feeds in Amiga files, S-E
Error	adds compile errors to a C file, S
Hello	window ex. from the RKM, S
Kermit	generic Kermit implementation, flakey, no terminal mode, S-E
Scales	sound demo plays scales, S-E
SkewB	Rubik cube demo in hi-res colors, S-E

AmigaBasicProgs(dir)

Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs
WitchingHour	a game

ABasiC programs:

Casino	games of poker, blackjack, dice, and craps
Gomoku	also known as 'othello'
Sabotage	sort of an adventure game

Executable programs:

Disassem	a 68000 disassembler, E-D
DpSlide	shows a given set of IFF pictures, E-D
Arrange	a text formatting program, E-D

Assembler programs:

Argoterm	terminal program with speech and Xmodem, S-E

AMICUS Disk 4 Files from the original Amiga Technical BBS

Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga technical support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

Complete and nearly up-to-date C source to 'image.ed', an early version of the Icon Editor. This is a little flaky, but compiles and runs.

An Intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemosll.h, nodos.c, and txwrite.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete ports
creatstd.c	create standard I/O requests
creatsak.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the 'dotty window' demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of 'freemap'
getbobs.c	tools for VSprites and BOBs
gfxmem.c	graphic memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers examples
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
seritest.c	tests serial port commands
serisamp.c	example of serial port use
prinintr.c	sample printer interface code
prtbase.h	printer device definitions
regintes.c	region test program
seiface.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
SetSerial.c	set the attributes (parity, data bits, etc) of the serial port
singplay.c	single playfield example
speechtoy.c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
tmrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all available system fonts
process.i and prtbase.i	assmebler include files:
autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console I/O chapter
diskfont.txt	warning of disk font loading bug
fullfunc.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter
License	information on Workbench distribution license
printer	pre-release copy of the chapter on printer drivers, from RKM 1.1
v1.1fd.txt	'diff' of .fd file changes from version 1.0 to 1.1
v28v1.diff	'diff' of include file changes from version 28 to 1.0

AMICUS Disk 5 Files from the Amiga Link / Amiga Information Network

Note that some of these files are old, and refer to older versions of the operating system. These files are from AmigaLink. For a time, Commodore supported Amiga Link, aka AIN, for online developer technical support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

A demo of Intuition menus called 'menudemo', in C source

whereis.c	find a file searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example

Assembler files:

mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i	
mydev.i	
asmsupp.i	
macros.i	assembler include files

Texts:

amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port spec
parallel	parallel port spec
serial	serial port spec
v1.1update	list of new features in version 1.1
v1.1h.txt	'diff' of include file changes from version

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk does contain a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The latest IFF spec is elsewhere in this library.

AMICUS Disk 6 IFF Pictures

This disk includes the DPSlide program, which can view a given series of IFF pictures, and the 'showpic' program, which can view each file at the click of an icon. The pictures include a screen from ArticFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny Martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Contruction Set, a TV newcaster, the PaintCan, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed.

AMICUS Disk 7 DigiView HAM demo picture disk

This disk has pictures from the DigiView hold-and-modify video digitizer. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and Robert. This includes a program to view each picture separately, and all together as separate, sizable screens. The 'seeibm' program, to turn any screen into an IFF picture.

AMICUS Disk 8

C programs:

Browse	view text files on a disk, using menus S-E-D
Crunch	removes comments and white space from C files, S-E
IconExec	EXECUTE a series of commands from Workbench S-E
PDScreen Dump	dumps Rastport of highest screen to printer
SetAlternate	sets a second image for an icon, when clicked once S-E
SetWindow	makes windows for a CLI program to run under Workbench S-E
SmallClock	a small digital clockin a window menu bar
Scrimper	the screen printer in the fourth AC S-E

Amiga Basic Programs:

(Note: Many of these programs are present on AMICUS Disk 1. Several of these were converted to Amiga Basic, and are included here.)

AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert Compuserve hex files to binary, S-D
Clue	the game, Intuition driven
ColorArt	art drawing program
DeluxeDraw	the drawing program in the 3rd AC, S-D
Eliza	conversational computer psychologist
Othello	the game, as known as 'go'
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Spelling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse

Executable programs:

3Dcube	Module-2 demo of a rotating cube
AltIcon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spell checker, E-D
arc	the ARC file compression progra must-have for telecom, E-D
Bertrand	graphics demo
disksalvage	prog. to rescue trashed disks, E-D
KwikCopy	a quick but nasty disk copy program: ignores errors, E-D
LibDir	lists hunks in an object file E-D
SaveILBM	saves any screen as IFF pic.E-D ??
ScreenDump	shareware screen dump prog, E only
StarTerm	version 2.0, term program, XmodemE-D

Texts:

LatticeMain	tips on fixing _main.c in Lattice
GDiskDrive	make your own 5 1/4 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of the MicroForge HD
PrintSpooler	EXECUTE-based print spool prog.

.BMAP files:

These are the necessary links between Amiga Basic and the system libraries. To take advantage of the Amiga's capabilities in Basic, you need these files. BMAPs are included for 'dist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', mathieeedoubas', 'mathieeesingbas', 'mathtrans', 'potgo', 'timer' and 'translator'.

AMICUS Disk 9

Amiga Basic Programs:

FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, & Intensity
Requester	ex. of requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilities
Synthesizer	sound program
WorldMap	draws a map of the world

Executable programs:

Boing!	latest Boing! demo,with selectable speed,E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time & date
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
Boing!	latest Boing! demo,with selectablespeed, E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time date,
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	explains how to read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
PrinterTip	sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of Transformer programs that work

Printer Drivers:

Printer drivers for the Canon PJ-1080A, the C Itoh Prowriter, an improved Epson driver that eliminates streaking, the Epson LQ-800, the Gemini Star-10, the NEC 8025A, the Okidata ML-92, the Panasonic KX-P10xx family, and the Smith-Corona D300, with a document describing the installation process.

AMICUS Disk 10 Instrument sound demos

This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, a banjo, a bass guitar, a boink, a calliope, a car horn, claves, water drip, electric guitar, a flute, a harp arpegio, a kickdrum, a marimba, a organ minor chord, people talking, pigs, a pipe organ, a Rhodes piano, a saxophone, a sitar, a snare drum, a steel drum, bells, a vibrophone, a violin, a wailing guitar, a horse whinny, and a whistle.

AMICUS Disk 11

C programs

drutil	Intuition-based, CLI replacement manager S-E
cpri	shows and adjusts priority of CLI processes, S-E
ps	shows info on CLI processes, S-E
victex	displays Compuserve RLE pics, S-E

AmigaBasic programs

pointered	pointer and sprite editor program
optimize	optimization ex ample from AC article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB OBJECTS
grids	draw and play waveforms
hilbert	draws Hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows3D	3D graphics program, from A C™ article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds

Executable programs

cp	unix-like copy command, E
cls	screen clear, S-E
diff	unix-like stream editor uses 'diff' output to fix files
pm	chart recorder performances indicator

Assembler programs

cls	screen clear and CLI arguments example

Modula-2

trails	moving-worm graphics demo
caseconvert	converts Modula-2 keywords to uppercase
Forth	Breshehan circle algorithm example
Analyze	12 templates for the spreadsheet Analyze

There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Getting the files from your C-64 to your Amiga is the hard part.

AMICUS Disk 12

Executable programs

blink	'alink' compatible linker, but faster, E-D
clean	spins the disk for disk cleaners, E-D
epsonset	sends Epson settings to PAR from menu E-D
showbig	view hi-res pics in low-res superbitmap, E-D
speaktime	tell the time, E-D
undelete	undeletes a file, E-D
cnvapldhm	converts Apple][low, medium and high res pictures to IFF, E-D
menued	menu editor produces C code for menus, E-D
quick	quick disk-to-disk nibble copier, E-D
quickEA	copies Electronic Arts disks, removes protection, E-D
txed 1.3	demo of text editor from Microsmiths,E-D

C programs

spin3	rotating blocks graphics demo, S-E-D
popdi	start a new CLI at the press of a button, like Sidekick, S-E-D
vsprite	VSprite example code from Commodore, S-E-D
AmigaBBS	Amiga Basic bulletin board prog., S-D

Assembler programs

star10	makes star fields like Star Trek intro,S-E-D

Pictures

Mount Mandelbrot	3D view of Mandelbrot set
Star Destroyer	hi-res Star Wars starship
Robot	robot arm grabbing a cylinder

Texis

vendors	Amiga vendors, names, addresses
cardco	fixes to early Cardco memory boards
cinclude	cross-reference to C include files
mindwalker	clues to playing the game well
slideshow	make your own slideshows from the Kaleidoscope disk

AMICUS Disk 13

Amiga Basic programs

Routines from Carolyn Scheppner of CBM Tech Support, to read and display IFF pictures from Amiga Basic. With documentation. Also included is a program to do screen prints in Amiga Basic, and the newest BMAP files, with a corrected ConvertFD program. With example pictures, and the SaveLBM screen capture program.

Routines to load and play FutureSound and IFF sound files from Amiga Basic, by John Foust for Applied Visions. With documentation and C and assembler source for writing your own libraries, and interfacing C to assembler in libraries. With example sound.

Executable programs

gravity	Sci Amer Jan 86 gravitation graphic simulation, S-E-D

Texts

MIDI	make your own MIDI instrument interface, with documentation and a hi-res schematic picture.

AMICUS Disk 14

Several programs from Amazing Computing issues:

Tools

Dan Kary's	C structure index program, S-E-D

Amiga Basic programs:

BMAP Reader	by Tim Jones
IFFBrush2BOB	by Mike Swinger
AutoRequester	example
DOSHelper	Windowed help system for CLI commands, S-E-D
PETrans	translates PET ASCII files to ASCII files, S-E-D
C Squared	Graphics program from Scientific American, Sept 86, S-E-D
crlf	adds or removes carriage returns from files, S-E-D
dpdecode	decrypts Deluxe Paint, remo
ves copy	protection, E-D
queryWB	asks Yes or No from the user returns exit code, S-E
vc	VisiCalc type spreadsheet, no mouse control, E-D
view	views text files with window and
slider	gadget, E-D
Cing, Sproing, ye	Boing, Zoing are sprite-based
Boing! style	demos, S-E-D

CLIClock, sClock, wClock are window border clocks, S-E-D

Texts

An article on long-persistance phospor monitors, tips on making brushes of odd shapes in Deluxe Paint, and recommendations on icon interfaces from Commodore-Amiga.

AMICUS 15

The C programs include:

'pr'	a file printing utility, which can print files in the background, and with line numbers and control character filtering.
'fm'	displays a chart of the blocks allocated on a disk.
'Ask'	questions an 'execute' file, returns an error code to control the execution in that batch file
'Stat'	an enhanced version of AmigaDOS 'status' command.
'Dissolve'	random-dot dissolve demo displays IFF picture slowly, dot by dot, in a random fashion.
'PopCLI2'	invoke new CLI window at the press of a key.

The executable programs include:

'Form'	file formatting program through the printer driver to select print styles
'DiskCat'	catalogs disks, maintains, sorts,merges lists of disk files
'PSound'	SunRize Industries' sampled sound editor & recorder
'Iconmaker'	makes icons for most programs
'Fractals'	draws great fractal seascapes and mountain scapes.
'3D Breakout'	3D glasses, create breakout in a new dimension
'AmigaMonitor'	displays lists of open files,
memory use,	tasks, devices and ports in use.
'Cosmoroids'	version of 'asteroids' for the Amiga.
'Sizzlers'	high resolution graphics demo written in Module 2.

Texts:

'ansi.txt'	explains escape sequences the CON: device responds to.
'FKey'	includes template for making paper to sit in the tray at the top of the Amiga keyboard.
'Spawn'	programmer's document from Commodore Amiga, describes ways to use the Amiga's multitasking capabilities in your own programs.

AmigaBasic programs:

'Grids'	draw sound waveforms, and hear them played.
'Light'	a version of the Tron light-cycle video game.
'MegaSol'	a game of solitaire.
'Stats'	program to calculate batting averages
'Money'	"try to grab all the bags of money that you can."

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS 16

'juggler'	demo by Eric Graham. a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

IFF pictures

parodies of the covers of Amiga World and Amazing Computing magazines.

C programs:

'inputhandler'	example of making an input handler.
'FileZap3'	binary file editing program
'ShowPrint'	displays IFF picture, and prints it.
'Gen'	program indexes and retrieves C structures and variables declared in the Amiga include file system.

Executable Programs:

'FixHunk2'	repairs an executable program file for expanded memory
'ms2smus'	converts Music Studio files to IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases.
'Missile'	Amiga version of the 'Missile Command' video game,

This disk also contains several files of scenerios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park

AMICUS 17

Telcommunications disk whuch contains six terminal programs.

"Comm" V1.33	term prog. with Xmodem, WXmodem,
"ATerm" V7.2	term prog. includes Super Kermit
"VT-100"V2.6	Dave Wecker's VT-100 emulator with Xmodem,Kermit, and scripting
"Amiga Kermit"	V4D(060) port of the Unix C-Kermit
"VTek" V2.3.1	Tektronix graphics terminal emulator based on the VT-100 prog. V2.3 and contains latest 'arc' file compression
"AmigaHost"	V0.9 for Compuserve. Includes RLE graphics abilities & CIS-B file transfer protocol.
"FixHunk"	expansion memory necessity
"FixObj"	removes garbage characters from modem received files
"Txt"	filters text files from other systems to be read by the Amiga E.C.
"addmem"	executeable version for use with mem expansion article in AC v2..1
'arc'	file documentation and a basic tutorial on un 'arc'ing files
"arcre"	for makeing "arc" files E.C.

AMICUS Disk 18

Logo	Amiga version of the populecomputer language, with example programs, E-D
Tv'Text	Demo version of the TV'Text character generator
PageSetter	Freely distributable versions of the updated PagePrint and PageIFF programs for the PageSetter desktop publishing package.
FullWindow	Resizes any CLI window using only CLI commands, E-D
Life3d	3-D version of Conway's LIFE program, E-D
Defdisk	CLI utility to re-assign a new Workbench disk, S-E-D
Calendar.WKS	Lotus-compatible worksheet that makes calendars
SetKey	Demo of keyboard key re-programmer, with IFF picture to make function key labels, E-D
VPG	Video pattern generator for aligning monitors, E-D
HP-10C	Hewlett-Packard-like calculator, E-D
SetPrefs	Change the Preferences settings on the fly, in C, S-E-D
StarProbe	Program studies stellar evolution. C source included for Amiga and MS-DOS, S-E-D
ROT	C version of Colin French's AmigaBasic ROT program from Amazing Computing. ROT edits and displays polygons to create three dimensional objects. Up to 24 frames of animation can be created and displayed. E-D
Scat	Like fng, windows on screen run away from the mouse, E-D
DK	Decays' the CLI window into dust, in Modula 2, S-E-D
DropShadow2	Adds layered shadows to Workbench windows, E-D

AMICUS 19

This disk carries several programs from Amazing Computing. The IFF pictures on this disk include the Amiga Wake part T-shirt logo, a sixteen-color hi-res image of Andy Griffith, and five Amiga Livel pictures from the Amazing Stories episode that featured the Amiga.

Solve	Linear equation solver in assembly language, S-E-D
Gadgets	Bryan Catley's AmigaBasicGadTorial,
Household	Bryan Catley's AmigaBasic household inventory program, S-D
Waveform	Jim Shields' Waveform WoAmigaBasic, S-D
DiskLib	John Kennan's AmigaBasic disk librarian program, S-D
Subscripts	Ivan Smith's AmigaBasic subscript example, S-D
String, Boolean	C programs and executables for Harriet Maybeck Tolly's Intuition tutorials, S-E-D
Skinny C	Bob Riemersma's example for making small C programs, S-E-D
COMAL.h	Make C look like COMAL HeaDer file,
EmacsKey	Makes Emacs function key definitions by Greg Douglas, S-D
AMon 1.1	Snoop on system resource use, E-D
BTE	Bard's Tale character editor, E-D
Size	CLI program shows the size of a given set of files, E-D
WinSize	CLI window utility resizes current window, S-E-D

Disk 20

Compector, Decoder Steve Michel AmigaBasic tools, S-D

BobEd	BOB and sprite editor written in C,S-E-D
SpriteMasterII	Sprite editor and animator by Brad Kiefer, E-D
BitLab	Blitter chip exploration C program by Tomas Rokicki, S-E-D
FPic	Image processing program by Bob Bush loads and saves IFF images, changes them with several techniques, E-D
Bankn	Complete home banking program, balance your checkbook! E-D
cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory
input.dev	sample input handler, traps key or mouse events
joystick	Shows how to set up the gameport device as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with ID requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

AMICUS 21

Target	Makes each mouse click sound like a gunshot, S-E-D
Sand	Simple game of sand that follows the mouse pointer, E-D
PropGadget	Harriet Maybeck Tolly's proportional gadget example, S-E
EHB	Checks to see if you have extra-half-bright graphics, S-E-D
Piano	Simple piano sound program
CelScripts	Makes cel animation scripts for Aegis Animator, in AmigaBasic

This disk has electronic catalogs for AMICUS disks 1 to 20 and Fish disks 1 to 80. They are viewed with the DiskCat program, included here.

AMICUS 22

Cycles	Light cycle game, E-D
Show_Print!	Views and prints IFF pictures, including larger than screen
PrtDrvGen2.3	Latest version of a printer driver generator
Animations	VideoScape animations of planes and boing ball
Garden	Makes fractal gardenscapes
BasicSorts	Examples of binary search and insertion sort in AmigaBasic

Fred Fish Public Domain Software

Fred Fish Disk 1:

amigademo	Graphical benchmark for comparing amigas.
amigaterm	simple communications program with Xmodem
balls	simulation of the "kinetic thingy" with balls on strings
colorful	Shows off use of hold-and-modify mode.
dhrystone	Dhrystone benchmark program.
dotty	Source to the "dotty window" demo on the Workbench disk.
freedraw	A small "paint" type program with lines, boxes, etc.
gad	John Draper's Gadget tutorial program
gtxmem	Graphical memory usage display prog.
halfbrite	demonstrates "Extra-Half-Brite" mode, if you have it
hello	simple window demo
latffp	accessing the Motorola Fast Floating Point library from C
palette	Sample prog. to design color palettes.
trackdisk	Demonstrates use of the trackdisk driver.
requesters	John Draper's requester tutorial and example program.
speech	Sample speech demo program. Stripped down "speechtoy".
speechtoy	Another speech demo program.

Fred Fish Disk 2:

alib	Object module librarian.
cc	Unix-like frontend for Lattice C compiler.
dbug	Macro based C debugging package. Machine independent.
make	Subset of Unix make command.
make2	Another make subset command.
microemacs	Small version of emacs editor, with macros, no extensions
portar	Portable file archiver.
xrf	DECUS C cross reference utility.

Fred Fish Disk 3:

gothic	Gothic font banner printer.
roff	A "roff" type text formatter.
ff	A very fast text formatter
cforth	A highly portable forth implementation. Lots of goodies.
xlisp	Xlisp 1.4, not working correctly.

Fred Fish Disk 4:

banner	Prints horizontal banner
bgrep	A Boyer-Moore grep-like utility
bison	CNU Unix replacement 'yacc', not working.
bm	Another Boyer-Moore grep-like utility
grep	DECUS grep
kermit	simple portable Kermit with no connect mode.
MyCLI	Replacement CLI for the Amiga. V. 1.0
mandel	A Mandelbrot set program, by Robert French and RJ Mical

Fred Fish Disk 5

cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory

input.dev	sample input handler, traps key or mouse events
joystick	Shows how to set up the gameport device as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with IO requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

Fred Fish Disk 6:

compress	like Unix compress, a file squeezer
dadc	analog clock impersonator
microemacs	upgraded version of microemacs from disk 2
mult	removes multiple occuring lines in files
scales	demos using sound and audio functions
setparallel	Allows changing parallel port parameters
setserial	Allows changing serial port parameters.
sortc	quicksort based sort program, in C
stripc	Strips comments and extra whitespace from C source

Fred Fish Disk 7:

This disk contains the executables of the game Hack V 1.0.1.

Fred Fish Disk 8:

This disk contains the C source to Hack on disk 7.

Fred Fish Disk 9:

moire	Draws moire patterns in black and white
MVP-FORTH	Mountain View Press Forth, version 1.00.03A. A shareware version of FORTH from Fantasia Systems.
proff	a more powerful text formatting program
setface	Program to toggle interlace mode on and off.
skewb	a rubic's cube type demo
sparks	moving snake Graphics demo

Fred Fish Disk 10:

conquest	An interstellar adventure simulation game
dehex	convert a hex file to binary
filezap	Patch program for any type of file.
fixobj	Strip garbage off Xmodem transferred files.
iff	Routines to read and write iff format files.
ld	simple directory program
ls	Minimal UNIX ls, with Unix-style wildcarding, in C
sq,usq	file squeeze and unsqueeze
trek73	Star Trek game
yachtc	Dice game.

Fred Fish Disk 11:

dpslide	slide show program for displaying IFF images with miscellaneous pictures

Fred Fish Disk 12:

amiga3d	Shows a rotating 3 dimensional solid "Amiga sign".
ArgoTerm	a terminal emulator program, written in assembler
arrow3d	Shows a rotating 3 dimensional wire frame arrow.
ld4	directory listing program
IconExec	
SetWindow	two progs for launching progs from Workbench, presently only works under CLI
SetAlternate	Makes an icon show a second image when clicked once
StarTerm	terminal emulator, with ASCII Xmodem, dialer, more.

Fred Fish Disk 13:

A Bundle of Basic programs, including:

Jpad	toybox	ezspeak
mandlebrot	xmodem	3dsolids
addbook	algebra	ror
amgseq1	amiga-copy	band
bounce	box	brickout
canvas	cardl	circle
colorcircles	Copy	cubes1
cutpaste	date	dogstar
dragon	draw	dynamictriangle
Eliza	exterm	filibuster
fractal	fscape	gomoku
dart	haiku	hal9000
halley	hauntedM	hidden
join	loz	mandel
menu	minipaint	mouse
Orthello	patch	pena
pinwheel	gbox	random-circles
Readme	rgb	rgbtest
Rord	sabotage	salestalk
shades	shapes	shuttle
sketchpad	specdart	speakspeech
speecheasy	spell	sphere
spiral	striper	superpad
suprshr	talk	terminal
termtest	tom	topography
triangle	wheels	xenos
xmostriper		

(note: some programs are Abasic, most are Amigabasic, and some programs are presented in both languages)

Fred Fish Disk 14:

amiga3d	update of #12, includes C source to a full hidden surface removal and 3D graphics
beep	Source for a function that generates a beep sound
dex	extracts text from within C source files
dimensions	demonstrates N dimensional graphics
filezap	update of disk 10, a file patch utility
gfxmem	update of disk 1, graphic memory usage indicator
gi	converts IFF brush files to Image struct, in Ctext.
pdterm	simple ANSI VT100 terminal emulator, in 80 x 25 screen
shell	simple Unix 'csh' style shell
termcap	mostly Unix compatible 'termcap' implementation.

Fred Fish Disk 15:

Blobs	graphics demo, like Unix 'worms'
Clock	simple digital clock program for the title bar
Dazzle	An eight-fold symmetry dazzler program. Really pretty!
Fish	double buffered sequence cycle animation of a fish
Monopoly	A really nice monopoly game written in AbasiC.
OkidataDump	Okidata ML92 driver and WorkBench screen dump program.
Polydraw	A drawing program written in AbasiC.
Polyfractals	A fractal program written in AbasiC.

Fred Fish Disk 16:

A complete copy of the latest developer IFF disk

Fred Fish Disk 17:

The NewTek Digi-View video digitizer HAM demo disk

Fred Fish Disk 18:

AmigaDisplay	dumb terminal program with bell, selectable fonts
Ash	Prerelease C Shell-like shell program, history, loops, etc.
Browser	wanders a file tree, displays files, all with the mouse
MC68010	docs on upgrading your Amiga to use a MC68010
Multidim	rotate an N dimensional cube with a joystick
PigLatin	SAY command that talks in Pig Latin
Scrimper	Screen image printer
Xlisp1.6	source, docs, and executable for a Lisp interpreter.

Fred Fish Disk 19:

BlackJack	text-oriented blackjack game
JayMinerSlides	Slides by Jay Miner, Amiga graphics chip designer, showing flowchart of the Amiga internals, in 640 x 400.
Keymap_Test	test program to test the key mapping routines
LockMon	Find unclosed file locks, for programs that don't clean up.

Fred Fish Disk 20:

AmigaToAtari	converts Amiga object code to Atari format
DiskSalv	program to recover files from a trashed AmigaDOS disk.
Hash	example of the AmigaDOS disk hashing function
Hd	Hex dump utility ala Computer Language magazine, April 86
MandelBrots	Mandelbrot contest winners
MultiTasking	Tutorial and examples for Exec level multitasking
Pack	strips whitespace from C source
PortHandler	sample Port-Handler program that performs. Shows BCPL environment clues.
Random	Random number generator in assembly, for C or assembler.
SetMouse2	sets the mouse port to right or left
SpeechTerm	terminal Emulator with speech capabilities, XModem
TxEd	Demo editor from Microsmith's Charlie Heath

Fred Fish Disk 21

This is a copy of Thomas Wilcox's Mandelbrot Set Explorer disk. Very good!

Fred Fish Disk 22

This disk contains two new "strains" of microemacs.

Lemacs	version 3.6 by Daniel Lawrence. For Unix V7, BSD 4.2, Amiga, MS-DOS, VMS. Uses Amiga function keys, status line, execute, startup files, more.
Pemacs	By Andy Poggio. New features include <ALT> keys as Meta keys, mouse support, higher priority, backup files, word wrap, function keys.

Fred Fish Disk 23

Disk of source for MicroEmacs, several versions for most popular operating systems on micros and mainframes. For people who want to port MicroEmacs to their favorite machine.

Fred Fish Disk 24:

Conques	interstaller adventure simulation game
Csh	update to shell on Disk 14, with built in commands,named variables substitution.
Modula-2	A pre-release version of the single pass Modula-2 compiler originally developed for Machtosh at ETHZ. This code was transmitted to the AMIGA and is executed on the AMIGA using a special loader. Binary only.

Fred Fish Disk 25

Graphic Hack	A graphic version of the game on disks 7 and 8 This is the graphics-oriented Hack game by John Toebes. Only the executable is present.

Fred Fish Disk 26

UnHunk	Processes the Amiga "hunk" loadfiles.

Collect code, data, and bss hunks together, allows individual specificatio of code, data, and bss origins, and generates binary file with format reminiscent of Unix "a.out" format. The output file can be easily processed by a separate program to produce Motorola "S-records" suitable for downloading to PROM programmer. By Eric Black.

C-kermit	Port of the Kermit file transfer program and server.
Ps	Display and set process priorities
Archx	Yet another program for bundling up text files and mailing or posting them as a single file unit

Fred Fish Disk 27

ABdemos	Amiga Basic demos from Caroly Scheppner.
NewConvertFD	creates .bmaps from fd files.
BitPlanes	finds addresses of and writes to bitplanes of the screen's bitmap.
AboutBmaps	A tutorial on creation and use of bmaps.
LoadILBM	loads and displays IFF ILBM pics.
LoadACBM	loads and displays ACBM pics.
ScreenPrint	creates a demo screen and dumps it to a graphic printer.
Disassem	Simple 68000 disassembler. Reads standard Amiga object files and disassembles the code sections. Data sections are dumped in hex. The actual disassember routines are set up to be callable from a user program so instructions in memory can be disassembled dynamically. By Bill Rogers.
DvorakKeymap	Example of a keymap structure for the Dvorak keyboard layout. Untested but included because assembly examples are few and far between. By Robert Burns of CA
Hypocycloids	Spirograph, from Feb. 84 Byte.
LinesDemo	Example of proportional gadgets to scroll a SuperBitMap.
MemExpansion	Schematics and directions for building your own homebrew 1 Mb memory expansion, by Michael Fellinger.
SafeMalloc	Program to debug 'malloc()' calls
ScienceDemos	Convert Julian to solar and sidereal time, stellar positions and radial velocity epoch calculations and Galilean satellite plotter. By David Eagle.

Fred Fish Disk 28

ABasic games by David Addison: Backgammon , Cribbage, Milestone, and Othello

Cpp	DECUS 'cpp' C preprocessor, and a modified 'cc' that knows about the 'cpp', for Manx C.
Shar	Unix-compatible shell archiver, for packing files for travel.
SuperBitMap	Example of using a ScrollLayer, syncing SuperBitMaps for printing, and creating dummy RastPorts.

Fred Fish Disk 29

AegisDraw Demo	Demo program without save and no docs.
Animator Demo	Player for the Aegis Animator files
Cc	Unix-like front-end for Manx C
Enough	Tests for existence of system resources, files, and devices
Rubik	Animated Rubik's cube program
StringLib	
Vt100	VT-100 terminal emulator with Kermit and Xmodem protocols

Fred Fish Disk 30

Several shareware programs. The authors request a donation if you find their program useful, so they can write more software.

BBS	an Amiga Basic BBS by Ewan Grantham
FineArt	Amiga art
FontEditor	edit fonts, by Tim Robinson
MenuEditor	Create menus, save them as C source, by David Pehrson
StarTerm3.0	Very nice telecommunications by J. Nangano

(Fred Fish Disk#30 is free if requested when ordered with at least three other disks from the collection.)

Fred Fish Disk 31

Life	Life game, uses blitter to do 19.8 generations a second.
Mandelbrot	Version 3.0 of Robert French's program.
MxExample	Mutual exclusion gadget example.
RamSpeed	Measure relative RAM speed, chip and fast.
Set	Replacement for the Manx "set" command for environment variables,with improvements.
Tree	Draws a recursive tree, green leafy type, not files.
TxEd	Crippled demo version of Microsmith's text editor, TxEd.
VDraw	Full-featured drawing program by Stephen Vermeulen.
Xicon	Invokes CLI scripts from icon
Ticon	Displays text files from an icon.

Fred Fish Disk 32

Address	Extended address book written in AmigaBasic.
Calendar	Calendar/diary program written in AmigaBasic.
DosPlus1	First volume of CLI oriented tools for developers.
DosPlus2	Second volume of CLI oriented tools for developers.

Executables only:

MacView	Views MacPaint pictures in Amiga low or high res, no sample pictures, by Scott Evemden.
Puzzle	Simulation of puzzle with moving square tiles.
ShowHAM	View HAM pictures from CLI
Solitaire	ABasiC games of Canfield and Klondike, from David Addison.
Spin3	Graphics demo of spinning cubes, double-buffered example.
Sword	Sword of Fallen Angel text adventure game written in Amiga Basic.
Trails	Leaves a trail behind mouse, in Modula-2

Fred Fish Disk 33

3dstars	3d version of the "stars" program below.
Bigmap	Low-level graphics example scrolls bitmap with ScrollVPort.
Dbuf.gels	Double-buffered animation example for BOBs and VSprites.
DiskMapper	Displays sector allocation of floppy disks.
MemView	View memory in real time, move with joystick.
Oing	Bouncing balls demo
Sproing	Oing, with sound effects.
ScreenDump	Dumps highest screen or window to the printer.
Sdb	Simple database program from a DECUS tape.
Stars	Star field demo, like Star Trek.
TermPlus	Terminal program with capture, library, function keys, Xmodem, CIS-B protocols.
Vt100	Version 2.0 of Dave Wecker's VT-100 emulator, with scripts & function

Fred Fish Disk 34

Alint	Support files for Gimpel's 'lint' syntax checker
Blink	PD 'alink' compatible linker,faster,better.
Browser	Updated to FF 16 'browser', in Manx, with scroll bars, bug fixes.
Btree	b-tree data structure examples
Btree2	Another version of 'btree'
Calendar	Appointment calendar with alarm.
Less	File viewer, searching, position by percent, line number.
NewFonts	Set of 26 new Amiga fonts from Bill Fischer
Pr	Background print utility, style options, wildcards.
Requester	Deluxe Paint-type file requester, with sample.

Fred Fish Fred Fish Disk 35

ASendPacket	C example of making asynchronous I/O calls to a DOS handler, written by C-A
ConsoleWindow	C example of getting the Intuition pointer a CON: or RAW: window, for 1.2, by C-A.
DirUtil	Walk the directory tree, do CLI operations from menus
DirUtil2	Another variant of DirutI.
FileRequester	Lattice C file requester module, with demo driver, from Charlie Heath.
MacView	Views MacPaint pictures in Amiga low or high res, with sample pictures, by Scott Evemden.
Plop	Simple IFF reader program
PopCLI	Sidekick-style program invokes a new CLI, with automatic screen blanking.
QuickCopy	Devenport disk copiers duplicate copy-protected disks.
ScrollPf	Dual playfield example, from C-A, shows 400 x 300 x 2 bit plane playfield on a 320 x 200 x 2 plane deep playfield.
SendPacket	General purpose subroutine to send AmigaDos packets.
SpriteMaker	Sprite editor, can save work as C data structure. Shareware by Ray Larson.
Tracker	Converts any disk into files, for electronic transmission. Preserves entire file structure. Shareware by Brad Wilson.
TriClops 3-D	space invasion game, formerly commercial, now public domain. From Geodesic Publications.
Tsize	Print total size of all files in subdirectories.
Unifdef	C preprocessor to remove given #ifdef'd sections of a file, leaving the rest alone. By Dave Yost
Vttest	VT-100 emulation test program. Requires a Unix system.

Fred Fish Disk 36

Acp	Unix-like 'cp' copy program
Clock	Updated version of clock on disk 15.
Csh	Manx 'csh'-like CLI, history, variables, etc.
DietAid	Diet planning aid organizes recipes, calories
Echo	Improved 'echo' command with color, cursor addressing
FixHunk	Fixs programs to let them run in external memory.
Fm	Maps the sectors a file uses on the disk.
KickBench	Docs, program to make a single disk that works like a Kickstart and Workbench.
Lex	Computes Fog, Fleach, and Kincaid readability of text files.
TunnelVision	David Addison ABasic 3D maze perspective game.
Vc	Visicalc-like spreadsheet calculator program.
Vt100	Version 2.2 of Dave Wecker's telecom program
YaBoing	Oing! style game program shows sprite collision detects

Fred Fish Disk 37

This disk is a port of Timothy Budd's Little Smalltalk system, done by Bill Kinnersley at Washington State University.

Fred Fish Disk 38

CSquared	Sep 86 Sci American, Circle Squared algorithm
FixObj	Strips garbage off Xmodem transfered object files
Handler	AmigaDOS handler (device) example from C-A
Hp-10c	Mimics a HP-10C calculator, written in Modula-2
IFFEncode	Saves the screen as an IFF file
IffDump	Dumps info about an IFF file
Jsh	BDS C-like CLI shell
NewStat	STATUS-like program, shows priority, processes
Reversi	Game of Reversi, version 6.1

UUdecode Translate binary files to text, Unix-like programs
Vdraw Drawing program, version 1.14
VoiceFiler DX MIDI synthesizer voice filer program
Window Example of creating a DOS window on a custom screen

Fred Fish Disk 39

AnsiEcho 'echo', 'touch', 'list', 'cls' written in assembler.
Display Displays HAM images from a ray-tracing program, with example pictures.
Driver Example device driver source, acts like RAM: disk
Xlisp XLisp 1.7, executable only

Fred Fish Disk 40

Ahost Terminal emulator with Xmodem, Kermit and CIS B protocols, function keys, scripts, RLE graphics and conference mode.
AmigaMonitor Dynamically displays the machine state, such as open files, active tasks, resources, device states, interrupts, libraries, ports, etc.
Arc Popular file compression system, the standard for transiting files
AreaCode Program that decodes area codes into state and locality.
Blink 'alink' replacement linker, version 6.5
Cosmo An 'asteriods' clone.
Dg210 Data General D-210 Terminal emulator
DirUtil Windowed DOS interface program, V 1.4
DOSHelper Windowed AmigaDOS CLI help program
PagePrint Prints text files with headers, page breaks, line numbers
PopCLI Starts a new CLI with a single keystroke, from any program, With a screen-saver feature. Versions2, with
SpriteEd Sprite Editor edits two sprites at a time
X-Spell Spelling checker allows edits to files

Fred Fish Disk 41

AmigaVenture Create your own text adventure programs in AmigaBasic.
Csh Version 2.03 of Dillon's C sh-like shell. Executable only
Dbug Macro based C debugging package, FF #2
DualPlayField example from CBM, update to Intuition manual
GetFile Heath's file requester, with source
LatXref Cross reference of Lattice 3.10 header files
Lines Line drawing demo program
SetFont Changes font used in a CLI window
Vt100 Version 2.3 of the VT-100 terminal program.

Fred Fish Disk 42

This disk contains an Amiga version of MicroGNUEmacs.

Fred Fish Disk 43

BasicBoing AmigaBasic program demos page flipping of a 3D cube
Bom Demo copy of B.E.S.T. Business Management System.
BosList A list of Amiga Bulletin Board Systems
Cc C compiler frontends for Manx and Lattice C
Copper A hardware copper list disassembler
InstIFF Converts Instruments demo sounds to IFF sampled sounds
PopColours Adjust RGB colors of any screen
SpriteClock Simple clock is displayed on a sprite above all screens
ST Emulator Non-serious Atari ST emulator
WBrun Lets Workbench programs be run from the CLI
Wild Two Unix shell style wild card matching routines

Fred Fish Disk 44

Icons Miscellaneous icons
NewIFF New IFF material from CBM for sampled voice and music files
RayTracePics The famous ray-tracing pictures, from FF#39, now converted to IFF HAM format for "much" faster viewing.
ViewILBM Displays normal and HAM ILBM files

Fred Fish Disk 45

Clue Clue board game
Make Another 'make', with more features
Pictures Miscellaneous pictures
Update Updates an older disk with newer files from another disk
Wheres Searches a disk for files of given name

Fred Fish Disk 46

Asm Shareware 68010 macro assembler, ROM Kernal Manual compatible
CheckModem 'execute' file program detects presence of modem
Egad Gadget editor from the Programmers Network
Jive Transforms a file from English to Jive.
MyJib A binary only copy of Ma tt's alternate runtime library. Author: Matt Dillon
ProffMacros Subset Berkeley 'ms' and 'mm' macros for 'proff'
ValSpeak Transforms a file from English to Valley Speak.

Fred Fish Disk 47

3D-Arm Simulation of a robotic a rm, very good graphics, teaching tool, including C source.
Juggler Eric Graham's stunning HAM animation of a robot juggler
VT-100 Version 2.4 of Dave Wecker's terminal emulator, with Xmodem and Kermit file transfer protocols

Fred Fish Disk 48

Bru Alpha version of a hard disk file archiver
Comm Version 1.30 of a terminal emulator with phone directories
Csh Version 2.04 of Matt Dillion's Unix 'csh'-like CLI replacement, including Lattice and Manx C source
Diskperf Disk benchmark program for Unix and Amiga
Du Computes disk storage of a file or directory
MemWatch Program to watch for programs that trash low memory. It attempts to repair the damage, and puts up a requester to inform you of the damage. From the Software Distillery.
Profiler A realtime execution profiler for Manx C programs. Includes C source.

Fred Fish Disk 49

Cycloids Update of electronic spirograph from disk 27
DirUtil Enhanced version of DirUtil from disk 35
MultiDef Scans a set of object modules and libraries searching for multiply defined symbols
MyUpdate Disk update utility with options for stripping comments from C header files, and interactive verification of the updating process
Plot Computes and displays 3 dimensional functions in hires
Polygon Moire type pattern generator with color cycling
QMouse Queries whether a mouse button is pressed. This can give a return code that can customize a startup-sequence based on whether a mouse button was pressed.
Touch Example of setting the datestamp on a file, using a technique from Commodore-Amiga
Trees More extensive version of the trees program on Disk 31

Fred Fish Disk 50

Asm Version 1.1 of a shareware 68000 macro assembler, compatible with the Metacomco assembler. This includes an example startup module and more Motorola mneumonics.
BreakOut A brick breakout game, uses 3-D glasses
DiskZap Version 1.1 of a program to edit disks and binary files
FirstSilicon A smart CLI replacement with full editing and recall of previous commands
Missile A Missile Command-type game, with sound, in assembler
PerfectSound Sound editor for a low-cost sound digitizer
Sizzlers Graphics demos
UnixArc Version of 'arc' for Unix System V machines, in C
Wombat Version 3.01 of Dave Warker's terminal emulator

Fred Fish Disk 51

Bison GNU for Unix 'yacc', working update to disk 4 version
Compress Update to the file compression program on Disk 6
Cos "Wheel of Fortune"-type game in AmigaBasic
DifSeed Unix-like 'dif' and 'sseq' for finding the differences between two files, and then recreating the other, given one file, and the list of differences.
Sq, Usq Portable versions of the CP/M squeeze and unsqueeze

Fred Fish Disk 52

Assign Replacement for AmigaDOS 'assign' command in C
Fractal Makes random fractal terrains
Poly, HAMPoly Workbench-type demos for making polygons in lores and HAM
MxGads Example of mutual exclusion gadgets with GadgetText
Tek4010 Tektronix 4010 terminal emulator
VDraw Versions 1.16 and 1.19 of a Deluxe Paint-like drawing program

Fred Fish Disk 53

Animations Demo animations with player program for Aegis Animator
ARCre Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and 'makedir'
Compiler Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet Update with source of the 'vc' spreadsheet on disk 36
TarSplit Port of program to split Unix 'tar' archives
UUencode Utilities to encode and decode binary files for ASCII transmission, expanding them by 35 percent Paint-like drawing program

Fred Fish Disk 53

Animations Demo animations with player program for Aegis Animator
ARCre Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and ' makedir'
Compiler Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet Update with source of the 'vc' spreadsheet on disk 36
TarSplit Port of program to split Unix 'tar'archives
UUencode Utilities to encode and decode binary files for ASCII transmission, expanding them by 35 percent

Fred Fish Disk 54

Hanoi Solves Towers of Hanoi Problem in it's own Workbench window. by Ali Ozer
ISpell Port of a Unix screen oriented, interactive speling checker. (Expansion RAM required) by Pace Willisson
Ing A Screen of lots of bouncing little windows by Leo 'Bois Ewhac' Schwab
Lav Displays number of tasks in run queue, averaged over last 1, 5, and 15 minute periods. by Willaim Rucklidge
MIDITools Programs to play/record through the MIDI I/F. by Fred Cassirer
MoreRows Program to make the Work Bench Screen larger than normal. by Neil Katin and Jim Mackraz
Tilt Program to make your Amiga look like it didn't pass vibration testing. by Leo 'Bois Ewhac' Schwab

Fred Fish Disk 55

Csh V2.05 of Matt Dillon's csh like shell (Modified for Manx C). by Matt Dillon, Modified by Steve Drew
NewStartups New C Startup modules:
AStartup.asm with 1.2 fixes and better quote handling.
TWStartup.asm opens a stdio window, using user specs. by Commodore, posted to BIX by Carolyn Schepper
Palette Change another program's screen colors. by Carolyn Schepper
PipeDevice Allows the standard output of one process to be fed to the standard input of another. by Matt Dillon
ScreenSave Save a normal or HAM mode screen as an IFF file. by Carolyn Schepper
ShanghaiDemo Demo of the Activision game Shanghai.
SoundExample A double buffered sound example for Manx C. by Jim Goodnow
Vsprites A working vsprite example. by Eric Cotton
Vt100 V2.6 of Dave's Vt100 terminal emulator with kermit and xmodem. by Dave Wecker

Fred Fish 56

ClipBoard Clipboard device interface routines, to provide a standard interface. by Andy Finkle
ConPackets Demos the use of DOS Packets, ConUnit, etc. by Carolyn Schepper
GetDisks Program to find all available disk device names and return them as an exec list. by Philip Lindsay
GetVolume Program to get volume name of the volume that a given file resides on. by Chuck McManis
Icon2C Reads an icon file and writes out a fragment of C code with the icon data structures. by Carolyn Schepper
MergeMem Program to merge the MemList entries of sequentially configured RAM boards. by Carolyn Schepper
mCAD An object oriented drawing program, V1.1 by Tim Mooney

Fred Fish 57

CutAndPaste Implementaions of Unix cut and paste commands. by John Weald
GraphIt Program to plot simple functions in 2 or 3 dimensions. by Flynn Fishman
Juggler V1.2 of robot juggler animation. Uses HAM mode and ray tracing. by Eric Graham
MouseReader Shareware program to read text files and view IFF files using only the mouse. by William Betz
Ogre Game of tactical ground combat in the year 2086. by Michael Caplinger; Amiga port by Hobie Orris
Splines Program to demonstrate curve fitting and rendering techniques. by Helene (Lee) Taran

Fred Fish 58

ASDG-rrd Extremely useful shareware recoverable ram disk. by Perry Kivolowitz
BigView Displays any IFF picture, independent of the physical display size, using hardware scroll. by John Hodgson
EGraph Reads pairs of x and y value from a list of files and draws a formatted graph. by Laurence Turner
HyperBase Shareware data management system. V1.5
MemClear Walks through the free memory lists, zeroing free memory along the way. by John Hodgson
NewZAP A third-generation multi-purpose file sector editing utility. V3.0 by John Hodgson
RainBow A Maurauder-Style rainbow generator. by John Hodgson
SMUSPlayers Two SMUS plays, to play SMUS IFF music formatted files. by John Hodgson
View A tiny ILBM viewer by John Hodgson
WBdump JX-80 optimized workbench printer that does not use DumpRPort. by John Hodgson

Fred Fish Disk 59

Browser Update to browser program on disks 18 and 34. S-E
Browser2 Another different browser program. E
Clock Clock program with fonts, colors. E
Dme Dillon text editor V1.22 for programmers. E-D
DropCloth Puts a pattern on the Workbench backdrop. E-D
DropShadow Puts shadows on Workbench windows. E-D
FixWB Similar to DropCloth, but doesn't work yet. S-D
mCAD Object-oriented drawing program, version 1.2.2. Much improved over disk 56.
Robotroff Demo of animated pointers on Workbench. S-E-D
Supermort General compounding/amortization loan calculator. E-D

Fred Fish Disk 60

Various shareware and freeware programs
Blitz Memory resident file viewer. Very fast. E-D
BlitzFonts Makes text output faster. E-D
HandShake Terminal emulator with VT52/VT100/VT102support. E-D
Med Mouse-driven text editor version 2.1. E-D
PrtDrvGen Generates printer drivers, version 1.1. Source available from author. E-D
Show Slideshow-like IFF viewer, V2.1. E-D
Uedit Customizable text editor V2.0. E-D
Ueturbo Example Uedit setup macros. S-E-D

Fred Fish Disk 61

ATPatch Patches Transformer to work under AmigaDOS 1.2. S-E-D
FillDisk Writes zeroes to free blocks on a disk for security. S-E-D
LPatch Patch for programs that abort when loading under AmigaDOS 1.2. S-E-D
MicroEmacs Conroy MicroEmacs V3.8b, newer than disk 22. S-E-D
PearlFont Like Topaz, but rounded edges.
Terrain Generates fractal scenery. S-E-D
VSprites Makes 28 Vsprites, from P&E Book.

Fred Fish Disk 62

This is a port of the Unix game 'Hack', by the Software Distillery, version 1.0.3D.

Fred Fish Disk 63

This is a port of the Unix game 'Larn', by the Software Distillery, version 12.0B.

Fred Fish Disk 64

This is an offical IFF specification disk from Commodore, an update to disk 16.

Fred Fish Disk 65

Awk Unix text processor, like 'awk'. Doesn't work, but source is included. S-E-D.
MWB Example of rerouting Workbench window open calls to another custom screen. Version 1.01, S-E-D
CloseWB Example for closing a custom Workbench screen. S-E-D
Cookie Generates one-line fortune-cookie aphorisms. S-E-D
JTime Build-your-own mouse port clock.
MenuBuilder Creates C source files for menus, based on text descriptions. S-E-D.
NewPackets CBM tutorial on new packets and structures in AmigaDos 1.2.
PascalToC Pascal to C translator, not so great. S-E-D
Prep 'ratfor'-like FORTRAN preprocessor. S-E-D
RunBack Starts programs from CLI, allowing CLI window to close. E-D
SunMouse This program automatically clicks in windows when the mouse is moved over them. Version 1.0, E-D

Fred Fish Disk 66

AmScsi Preliminary plans for a SCSI disk controller board.
Asm68k Macro assembler, version 1.0.1. E-D
Assigned Example for avoiding DOS insert-disk requester, by scanning the list of 'assigned names. S-E-D
Dk Pretends to eat away at CLI window. S-E-D
Flip Flips whole screen as a joke. S-E-D
Foogol Foogol cross-compiler generates VAX assembly code. S-E-D
Free Prints amount of free space on all drives. S-E-D
MallocTest malloc/free memory test program. S-E-D
Melt Pretends to melt the screen. S-E-D
Nart Graphic flying string demo. S-E-D
Purty Easy way to set printer attributes from Workbench. E-D
RayTracer Simple ray tracing program. E-D
SendPackets Updated CBM examples of packet routines on disk 35. S-E-D
SnapShot Memory resident screen dump. E-D
TagBBS Shareware BBS system, version 1.02.

Fred Fish Disk 67

AmCat Shareware disk cataloging program.
AmigaSpell Shareware Intuition spelling checker, V2.0. E-D
Bouncer 3-D bouncing ball written in MultiForth, S-E-D
Comm Terminal program version 1.33, E
Dux5 Another version of DirUtil. S-E-D
HexCalc Hex, octal, & decimal calculator. E-D
Icons Various big and alternate image icons.
Mandala Mandala graphics and sound. E
PersMeit Demo shareware personal file manager.
RSLClock Menu bar clock version 1.3. E-D
RTCubes Graphics demo of 3D cubes. E-D
Wheel "Wheel of Fortune"-type game, in AmigaBasic

Fred Fish Disk 68

This is version MG 1b of the MicroGNUEmacs. Source and executable ereincluded, as well as source for other computers besides the Amiga.

Fred Fish 69

Asm68k Macro assembler, v1.0.3, E-D
BitLab Bitter exploring program, in C, S-E-D
Conman Replacement console device handler adds editing and history to any application that uses CON:, v0.9, E-D
Console Replacement console routines, in C, S-E-D
Dk Decays the screen bit by bit, update to disk 66, in Modula-2, S-E-D
Frags Displays memory fragmentatibp listing the size of free memory blocks, in C, S-E-D
IconType Change the type of an icon, in C, S-E-D
Make 'make' in Manx C, S-E-D
MonProc Monitors processes for packet activity, in C,S-ED
MouseClock Turns mouse pointer into a digital clock,in C,SED
Sb Browses system structures, from Transactor magazine, v1.0, in C, S-E-D

Spew	Generates 'National Enquirer'-type headlines from rules file. In C,S-E-D
Spool	Three programs to demonstrate multitasking and spooling in a printer spooler. In C, v1.2, S-E-D
Wc	Counts words ala Unix 'wc', but faster, in C,S-ED

Fred Fish 70
This is a disk of shareware programs.

AmigaMonitor	Explores state of the system, v1.13
Arc	Standard file compressor andl librarian, v0.23, a port of MS-DOS v5.0. E-D
BlackBook	Phone book program.
DoTil	Intuition-driven file manipulator program, v2.0.
GravityWars	Game of planets, ships and black holes, v1.03.
Jobs	Alternate user interface to CLI and Workbench, v2.1.
Lens	Magnifies area around mouse, shows it in a window, v1.0.
Life-3d	3D version of the classic cellular-automaton game, v1.2.
Logo	Logo language interpreter
SetKey	Demo keymap editor, v1.0
Vpg	Makes displays for aligning video monitors, v1.0.

Fred Fish 71

AirFoil	Makes airfoils using the Joukowski transformation, in C, S-E-D
Amiga Basic	Miscellaneous programs including 3D plot program, a kaleidoscope, C-A logo drawing program file comparison utility string search program, S-E-D
Blocks	A variation of "lines", but with variable color blocks. E-D
Comm	Great terminal program, v1.34, E-D
DiskX	Utility for exploring file system.E-D
Fpic	Simple image processing program that operates on IFF pictures, with several filters, merging images, E-D
IconMk	Makes icons for files , v1.2a, E-D
Icons	New icons
NewFonts	Two new fonts; 'shalt18', an electronic ciruit element font, and 'tbm5', a PC-like font.
PetCLI	An AmigaBASIC CLI shell program.
PWDemo	Demo of the commercial product PowerWindows,v1.2. It aids creation of custom windows, menus, and gadgets, giving C or assembly source. E-D
Rot	Creates and animates 3-D objects, v0.5, E-D
TimeSet	Sets time from Workbench, E-D

Fred Fish 72
This is a disk of IFF pictures.

Fred Fish 73

Add	Customizes existing program menus with Amiga-key shortcuts. Also includes 'until', which waits until a given window is created. Shareware, in C, S-E-D.
AutoIconOpen	Fools WB into thinking mouse has double-clicked icons. In C, S-E-D
Dio	Generic Exec device interface code for opening libraries, getting multiple I/O channels, asynchronous operations, etc. In C, S-E-D.
Dissolve	Slowly displays IFF files, ala Nov 86 Dr. Dobb's program. In C, S-E-D
DTerm	Flexible, reprogrammable terminal program v1.10, E-D
Expose	Re-arranges windows so that at least one pixel of menu bar gadgets are exposed. In C, S-E-D.
Lit	Scans a text file, converts to C-style printable strings.C,v2.0, S-E-D
Lmv	"Long Movie", program views series of IFF picts in quick succession, upto 19 fps. Shareware, E-D
MouseOff	Mouse pointer disappears after ten seconds of non-use. In C, S-E-D
ParOut	Examples of controlling parallel port with resources instead of the PAR: device.In C,S-E-D
PenPalFont	Child-like font.
RunBackGround	Similar to RunBack on disk 66, runs program from the CLI allowing the CLI window to close. In C,S-E-D
SnapShot	Screendump utility,update FF 66.E-D
TypeAndTell	Example installs a device handler before Intuition, and speaks each key as it is pressed. In C and assembler, S-E-D
Xplor	Prints info about system lists, in assembler,S-E-D

Fred Fish Disk 74

Cled	Edits and recalls CLI commands, v1.3, E-D
Control	Intercepts graphic printer dump calls and accesses color map, width,and screen resolution. C,S-E-D
Dme	Simple WYSIWYG text editor for programmers,v1.25. Update of FF 59.E-D
DropShadow	Workbench dropshadows, v2.0. Update to disk 59. E-D
Funds	AmigaBASIC program tracks mutual or stock p-D
Less	Text viewing program, like Unix "more", v1.1, update to disk 34. S-E-D
Makemake	Scans C source files and constructs a vanilla 'makefile' in the current directory. S-E-D
mCAD	Object-oriented drawing prog. v1.2.4, update to FF 59.Shareware, E-D
Random	Simple random number generator in C. S-E-D
TDebug	Monitors devices by intercepting Exec SendIO() and DoIO() vectors, in C, v1.0, S-E-D
Units	Converts measurements in different units, includes "chart" option, in C, S-E-D
XCopy	Replacement for AmigaDOS 'copy', doesn't change the date, uses Unix wildcards. E-D

Fred Fish Disk 75

Bezier	Play with Bezier curves points and granularity, S-E-D
BSplines	Play with b-splines, as above, S-E-D
Comm	C source for Comm terminal program v1.34. S-E-D
Copy	Replacement 'copy' command v1.0, preserves date, in C, S-E-D
Diff	Simple 'diff' in C, S-E-D
DuM2	Another DirUtil in Modula-2, v1.5, S-E-D
Eless	Fast 'dir' program in C, S-E-D
Fd	Faster 'eless' in C, S-E-D
HardCopy	Sends a transcript of a CLI session to a file, in C, S-E-D
MouseOff	Update to disk 73, turns off mouse pointer, S-E-D
SetFont	Changes the font in a Workbench screen, v2.0, S-E-D
SpeedDir	Another fast 'dir', in assembler, S-E-D

Fred Fish Disk 76 & 77
These are disks 1 and 2 of Chris Gray's Draco distribution for the Amiga. Draco is a compiled, structured language reminiscent of both C and Pascal. A full interface to AmigaDOS and Intuition is supplied. Be sure to get both disk 76 and 77.

Fred Fish Disk 78

Cycles	Cycle game like "Tron", v1.0, E-D
EOMS	Experts Only Mercenary Simulator game, E-D
MandelVroom	Mandelbrot generator with enhanced palette controls, fixed/floating point, presets, v1.50, in Manx C, S-E-D

Fred Fish Disk 79

AsmTools	CLI tools in assembler: echo, loadit, mounted, setlace, why; S-E-D
AssignDev	Give devices multiple names, in C, S-E-D
AuxHandler	Example of a dos handler that allows use of a CLI via the serial port. Includes source. Author: Steve Drew
Cmd	Redirects printer output to a file, in C, S-E-D
Info	AmigaDOS 'info' replacement, in C and assembler, S-E-D
Kill	Removes a task and its resources, in C, S-E-D
M2Error	Displays errors from TDI Modula-2 compiles, S-E-D
MonProc	Update to process packet program from disk 69, in C, S-E-D
Mounted	Program for testing if a drive is present, in a script in C, S-E-D
Nro	Another 'roff'-style text formatter, in C, S-E-D
ParTask	Finds parent task, in C, S-E-D
QueryAny	For scripts, asks a question, accepts Y/N, gives return code. In assembler, S-E-D
ScnSizer	Resets preferences settings for screen size, in C, S-E-D
SharedLib	Example of a shared library, in C and assembler, S-E-D
Task	Simple CreateTask() example in C, S-E-D
Uw	Unix Windows client v1.0, in C, S-E-D
Who	Lists tasks on ready and wait queues, in C, S-E-D

Fred Fish Disk 80
Fred Fish 80 has been withdrawn due to copyright problems.

Fred Fish Disk 81

Asm68k	V1.1.0 of a macro assembler
AutoFacc	Shrinks the FACC window and moves it to the back
Brushes	53 custom IFF brushes of electronic symbols
CheckIFF	Checks structure of an IFF file CledV1.4 update to disk 74 of a simple command line editor
Conman	Replaces console handler to add editing and history to many programs
Fonts	Miscellaneous fonts
Icon	V6.0 of the icon programming language
KeyLock	Freezes the keyboard and mouse until pass word entered.
ScatDisplay	hack created from "fng"
Smush	Smushes an IFF file.
Target	Each mouse clikc becomes a gunshot

Fred Fish Disk 82

AdventureA	port of the classic Crowther and Woods Adventure game
AmicTerm	V0.50 of a telecommunications program, with scripts, redial, beeps, enhanced file requester
D2D-Demo	Demo version of Disk-2-Disk from Central Coast Software
DX-Synth	Voice filer program for Yamaha DX series synthesizers, update to disk 38
DiskMan	V1.0 of another DirUtil program
Icons	Miscellaneous new icons
Perl	Universal MIDI patch panel, v1.2
Rocket	Another Workbench hack, plays Lunar Lander
Sand	Game of sands following your pointer.

Fred Fish Disk 83
This disk contains a demo version of TeX from N Squared. It is limited to small files, and the previewer can only display ten pages or less, and only a small number of fonts are provided.

Fred Fish Disk 84

AudioToolsPrograms	from Rob Peck's July/August Amiga World article
BlitLab	Blitter experimentation program, V1.2, update to disk 69
Ed	Simple editor, similar to Unix 'ed', based on the editor in Software Tools.
GravityWars	Game of planets, ships and black holes, v1.04, update to disk 70.
HunkPad	Adds legal padding to executables for Xmodem transmission.
PipeHandler	An AmigaDOS pipe device which supports named pipes and taps. V1.2
PopCLI	V3.0 of a hot-key to invoke a CLI window, with screen blanker, update to disk 40.
Requester	Update to disk 34 of a file requester similar to DPaint.
ScottDevice	V33.1 of a 'mountable MicroForge SCSI driver.
Viacom	Another Schwab hack, makes TV-like static on screen.

Fred Fish Disk 85

Csh	V2.05 of Dillon's 'csh'-like shell
FileReq	Source to wildcard file requester
Hide	Hides expansion memory from programs
ImageTools	Shareware tools to manipulation IFF images
LowMem	ServerShared library to aid in low memory situations
Plot6	A star plotting program with source.
RawIO	Example of setting raw mode on standard input
Rocket	Lunary Lander for Workbench, with source.
VMore	'more'-like text viewing utility, v1.0 with source.
Vnews	Simple Unix news reader.

Fred Fish Disk 86

AutoPoint	Auto-selects window under the mouse pointer, with screensaver.
ClickToFront	Double-clicks in window brings it to front, v1.1, S-E-D
Cmd	V3.0 of a tool to redirect printer output to a file.
FileISG-Demo	Demo of Softwood File Isg, a database manager with sound and graphics.

Fred Fish Disk 87

AdvSys	Adventure system from Byte May 1987, v1.2 E-D
AutoIconOpen	Fools Workbench to open disk icons, V1.2 update to disk 73, S-E-D
Claz	Converts IFF files to PostScript, V2.0, S-E-D
Commodi	tesMackraz's Commodities Exchange, an exec library to manage the input handler, v0.4
Diff	Update to disk 75 of Unix-like 'diff', S-E-D
Dme	V1.27 of Dillon's text editor, update to disk 74 , E-D
DropShadow	V2.0 of program that puts shadows on Workbench, S-E-D
Elib	Shared library example in Manx C.
ID-Handler	An AmigaDOS device handler that generates unique identifiers, V1.0, S-E-D
Install	Alternate AmigaDOS 'install' programs, S-E-D
MemWatch	Waits for low memory trashing, V2.0, S-E-D
MovePointer	Moves pointer to given location, S-E-D
MoveWindow	Move window to given location, S-E-D
MunchingSq	Munching Squares hack, S-E-D
PalTest	Example shows test to see if this is a PAL machine, S-E-D
Sc	Generates random scenery, S-E-D
Tek4595	Tek4695 printer driver
WBDualPF	Example of dual-playfield screen, update to disk 41, S-E-D
WarpText	Fast text rendering routines, S-E-D
Yaiff	rExample IFF reader, S-E-D
Zoo	A file archiver like "arc", v1.42A, E-D

Fred Fish Disk 88
Fred Fish Disk 88 has been removed due to copyright problems

To Be Continued.....

In Conclusion

To the best of our knowledge, the materials in this library are freely distributable. This means they were either publicly posted and placed in the Public Domain by their Author, or they have restrictions published in their files to which we have adhered. If you become aware of any violation of the author's wishes, please contact us by mail.

•AC•

TO Order Public Domain Software, please use the form on the following page.

Piracy & Undue Harm

Unfortunately, someone in the Amiga software community feels it helpful to remove copyright notices and publisher's names from software and place these programs in the Public Domain.

This situation is a shame, since Fred Fish saw two such programs, and felt they would be useful additions to his collection. Not knowing the software's origin, Mr. Fish released disks 80 and 88 with these programs.

Of course, the original developer was upset and contacted Mr. Fish. Mr. Fish immediately stopped production, but is still waiting notification of possible criminal charges.

In short, the name of a respected Amiga contributor has been tarnished. The Fred Fish collection also now has two "holes" which will require explanation to new users for as long as the collection remains. Most important, a good friend and colleague faces possible criminal charges and penalties for a crime over which he had little or no control.

Software Piracy is a crime! It hurts eveyone. If this has never been clear before, there should be no doubt now. •AC•

Volume 1 Number 1 Premiere February 1986

Super Spheres By Kelly Kauffman An ABasic Graphics prog.
Date Virus By J Foust A disease may attack your Amigal
EZ-Term by Kelly Kauffman An ABasic Terminal program
Miga Mania by P. Kivolowitz Programming fixes & mouse care
Inside CLI by G. Musser a guided insight into the AmigaDos™
CLI Summary by G. Musser Jr. A list of CLI commands
AmigaForum by B. Lubkin Visit Compuserve's Amiga SIG
Commodore Amiga Development Program by D. Hicks
Amiga Products A listing of present and expected products

Volume 1 Number 2 March 1986

Electronic Arts Comes Through A look software from EA
Inside CLI: part two G. Musser George investigates CLI & ED
A Summary of ED Commands
Live! by Rich Miner A review of the Beta version of the Livel f
Online and the CTS Fabite 2424 ADH Modem review by J. Foust
Superterm V 1.0 By K. Kauffman A term. prog. in Amiga Basic
A Workbench "More" Program by Rick Wirch
Amiga BBS numbers

Volume 1 Number 3 April 1986

Analyze! a review by Ernest Viveros
Reviews of Racter, Baratacas and Mindshadow
Forth! The first of our on going tutorial
Deluxe Draw!! by R. Wirch An Amiga Basic art program
Amiga Basic, A beginners tutorial
Inside CLI: part 3 by George Musser George gives us PIPE

Volume 1 Number 4 May 1986

SkyFox and Articfox Reviewed
Build your own 5 1/4 Drive Connector By Ernest Viveiros
Amiga Basic Tips by Rich Wirch
Scrimper Part One by P. Kivolowitz prog to print Amiga screen
Microsoft CD ROM Conference by Jim O'Keane
Amiga BBS Numbers

Volume 1 Number 5 1986

The HSI to RGB Conversion Tool
by S. Pietrowicz Color manipulation in BASIC
AmigaNotes by Rick Rae The first of the Amiga music columns
Sidecar A First Look by John Foust A first "under the hood" look
John Foust Talks with R. J. Mical at COMDEX™
How does Sidecar affect the Transformer
an interview with Douglas Wyman of Simile
The Commodore Layoffs by J. Foust A look Commodore "cuts"
Scrimper Part Two by Perry Kivolowitz
Marauder reviewed by Rick Wirch
Building Tools by Daniel Kary

Volume 1 Number 6 1986

Temple of Apshai Triology reviewd by Stephen Pietrowicz
The Halley Project: A Mission in our Solar System
reviewed by Stephen Pietrowicz
Flow: reviewed by Erv Bobo
Textcraft Plus a First Look by Joe Lowery
How to start your own Amiga User Group by William Simpson
Amiga User Groups
Mailing List by Kelly Kauffman a basic mail list program
Pointer Image Editor by Stephen Pietrowicz
Scrimper: part three by Perry Kivolowitz
Fun With the Amiga Disk Controller by Thom Sterling
Optimize Your AmigaBasic Programs for Speed by Pietrowicz

Volume 1 Number 7 1986

Aegis Draw: CAD comes to the Amiga by Kelly Adams
Try 3D by Jim Meadows an introduction to 3D graphics
Aegis Images/ Animator: a review by Erv Bobo
Deluxe Video Construction Set reviewed by Joe Lowery
Window requesters in Amiga Basic by Steve Michel
ROT by Colin French a 3D graphics editor
"I C What I Think" Ron Peterson with a few C graphic programs
Your Menu Sir! by B Catley programming menues Amiga Basic
IFF Brush to AmigaBasic 'BOB' Basic editor by M Swinger
Linking C Programs with Assembler Routines on the Amiga
by Gerald Hull

Volume 1 Number 8 1986

The University Amiga
By G.Gamble Amiga's inroads at Washington State University
MicroEd a look at a one man army for the Amiga
MicroEd, The Lewis and Clark Expedition reviewed by Frizelle
Scribble Version 2.0 a review
Computers in the Classroom by Robert Frizelle
Two for Study by Robert Frizelle
review of Discovery and The Talking Coloring Book
True Basic reviewed by Brad Grier
Using your printer with the Amiga
Marble Madness reviewed by Stephen Pietrowicz
Using Fonts from AmigaBasic by Tim Jones
Screen SaVer by P. Kivolowitz A monitor protection prog. in C
Lattice MAKE Utility reviewed by Scott P. Evernden
A Tale of Three EMACS by Steve Poling
.bmap File Reader in Amiga Basic by T Jones

Volume 1 Number 9 1986

Instant Music Reviewed by Steve Pietrowicz
Mindwalker Reviewed by Richard Knepper
The Alegra Memory Board Reviewed by Rich Wirch
TxEd Reviewed by Jan and Cliff Kent
Amazing Directory A guide to the sources and resources
Amiga Developers A listing of Suppliers and Developers
Public Domain Catalog A listing of Amicus and Fred Fish PDS
Dos 2 Dos review R. Knepper
Transfer files from PC/MS-DOS and AmigaBasic
MaxiPlan review by Richard Knepper The Amiga Spreadsheet
Gizmoz by reviewed by Peter Wayner Amiga extras!
The Loan Information Program by Brian Catley
basic prog. to for your financial options
Starting Your Own Amiga Related Business by W. Simpson
Keep Track of Your Business Usage for Taxes by J. Kummer
The Absoft Amiga Fortran Compiler reviewed by R A. Reale
Using Fonts from AmigaBasic, Part Two by Tim Jones
68000 Macros on the Amiga by G. Hull Advance your ability.
TDI Modla-2 Amiga Compiler by Steve Faiwiszewski
Looking at an alternative to C

Volume 2 Number 1 1987

What Digi-View Is... Or, What Genlock Should Be! by J. Foust
AmigaBasic Default Colors by Bryan Catley
AmigaBasic Titles by Bryan Catley
A Public Domain Modula-2 System reviewed by Warren Block
One Drive Compile by Douglas Lovell
Using Lattice C with a single drive system
A Megabyte Without Megabucks by Chris Irving
An internal Megabyte upgrade
Digi-View reviewed by Ed Jakober
Defender of the Crown reviewed by Keith Conforti
Leader Board reviewed by Chuck Raudonis
Roundhill Computer System's PANEL reviewed by Ray Lance
Digi-Paint.....by New Tek previewed by John Foust
Deluxe Paint II ....from Electronic Arts previewed by John Foust

Volume 2 Number 2 1987

The Modem by Josph L. Rothman efforts of a BBS Sysop
MacroModem reviewed by Stephen R. Pietrowicz
GEMINI or "It takes two to Tango" by Jim Meadows
Gameing between machines
BBS-PC! reviewed by Stephen R. Pietrowicz
The Trouble with Xmodem by Joseph L. Rothman
The ACO Project....Graphic Teleconferencing on the Amiga
by S. R. Pietrowicz
Flight Simulator II....A Cros Country Tutorial by John Rafferty
A Disk Librarian in AmigaBASIC by John Kennan
Creating and Using Amiga Workbench Icons by Celeste Hansel
AmigaDOS version 1.2 by Clifford Kent
The Amazing MIDI Interface build your own by Richard Rae
AmigaDOS Operating System Calls and
Disk File Management by D. Haynie
Working with the Workbench by Louis A. Mamakos Prog in C

Volume 2 Number 3

The Amiga 2000™ by J Foust
A First look at the new high end of the Amiga™ line.
The Amiga 500™ by John Foust
A look at the new low priced Amiga
An Analysis of the New Amiga PCs by J. Foust
Speculation on the New Amigas
Gemini Part II by Jim Meadows
The concluding article on two-player games
Subscripts and Superscripts in AmigaBASIC by Ivan C. Smith
The Winter Consumer Electronics Show by John Foust
AmigaTrix by W. Block
Those little shortcuts that make using the Amiga™ easier
Intuition Gadgets by Harriet Maybeck Tolly
A journey through gadget-land using C
Shanghai reviewed by Keith M. Conforti
Chessmaster 2000 & Chessmate reviewedby Edwin V. Apel, Jr.
Zing! from Meridian Software reviewed by Ed Bercovitz
Forth! by Jon Bryan Get stereo sound into your Forth programs.
Assembly Language on the Amiga™ by Chris Martin
Roomers by theBandito Genlocks are finally shipping, & MOREIII
AmigaNotes by R. Rae Hum Busters... "No stereo? Y not?..."
The AMICUS Network by J. Foust
CES, user group issues and Amiga Expo"

Volume 2 Number 4 1987

Amazing Interviews Jim Sachs by S. Hull Amiga Artist
The Mouse That Got Restored by Jerry Hull and Bob Rhode
Mouse repair
Sluething Public Domain Disks with CLI by John Foust
using CLI on PDS
Highlights from the San Francisco Commodore Show
by Steve Hull
Speaker Sessions at the San Francisco Commodore Show
by Harriet Tolly
The Household Inventory System in AmigaBASIC™
by Bryan Catley
Secrets of Screen Dumps by Natkun Okun
Using Function Keys with MicroEmace by Greg Douglas
Amigatrix II by Warren Block
More shortcuts to make the Amiga work easier
Basic Gadgets by Brian Catley Create your own gadget functions
Gridiron reviewed by K. Conforti Real football for the Amiga
Star Fleet I Version 2.1 reviewed by J. Tracy Space in the Amiga
The TIC reviewed by John Foust
A tiny battery powered Clock Calendar
Metascope review by H. Tolly An easy-to-use debugger

Volume 2 Number 5

The Perfect Sound Digitizer by R. Battle review of SunRize's SD
The Future Sound Digitizer by W. Block review
of Applied Vision's SD
Forth! by J. Bryancomparing JForth amd Multi-Forth.
Basic Input by B. Catley Build your own input routine for use in
all your AmigaBASIC programs.
Writing a SoundScape Module in C by T. Fay Programming with
MIDI, Amiga and SoundScape with the author of SoundScape.
Programming in 68000 Assembly Language by C. Martin
A continuing discussion of Counters and Addressing Modes.
Using FutureSound with AmigaBASIC by J. Meadows
AmigaBASIC Programming utility with real, digitized STEREO
AmigaNotes by R. Rae
review of Mimetics' SoundScape Sound Sampler.
More AmigaNotes by R. Rae
A further review of Sunrize's Perfect Sound Digitzer.
Waveform Workshop in AmigaBASIC by J. Shields A utility to
edit and save waveform for use in other AmigaBASIC programs.
The Mimetics Pro MIDI Studio by Sullivan, Jeffery
A review of Mimetics' music editor/player.
Intuition Gadgets Part II by H. MaybeckTolly, Boolean gadgets
provide the user with an on/off user interface.

Volume 2 Number 6

Forth! by J. Bryan Access the huge resources in the ROM Kernal.
The Amazing Computing Hard Disk Review
by J. Foust & S. Leemon In-depth looks at the C Ltd. Hard
Drive, Microbotics' MAS-Drive20, Byte by Byte's PAL Jr.,
Supra's 4x4 Hard Drive and Xebec's 9720H Hard Drive. Also. a
look at disk driver software currently under devlopment.
Modula-2 AmigaDOS™ Utilities by S. Falwiszewsk
A demonstration of calls to AmigaDOS and the ROM kernal.
Amiga Expansion Peripheral by J. Foust
Explanation of Amiga expansion peripherals.
Amiga Technical Support by J. Foust
How and where to get Amiga tech support.
Goodbye Los Gatos by J. Foust Closeing AmigaCorp Los Gatos.
The Amicus Network by J. Foust West Coast Computer Faire.
Metacomco Shell and Toolkit by J. Foust A review
The Magic Sac by J. Foust Run Mac programs on your Amiga.
What You Should Know Before Choosing an Amiga 1000
Expansion
Device by S. Grant
7 Assemblers for the Amiga by G. Hull Choose your assembler
High Level Shakeup Replaces Top Management at
Commodoreby S. Hull
Peter J. Baczor"by S. Hull Manager at CBM'sgives an inside look
Logistix A review by Richard Knepper
Organize! by Richard Knepperdatabase.
68000 Assembly Language Programming on the Amiga
by Chris Martin
Superbase Personal Relational Database by Ray McCabe
AmigaNotes by Rae, RichardA look at FutureSound
Commodore Shows the Amiga 2000 and 500 at
the Boston Computer Society by Harriet Maybeck Tolly

Volume 2, Number 7

New Breed of Video Products by John Foust...
Very Vivid! by Tim Grantham...
Video and Your Amiga by Oran Sands III
Amigas & Weather Forecasting by Brenden Larson
A-Squared and the Live! Video Digitizer by John Foust.
Aegis Animator Scripts and Cel Animation by John Foust
Quality Video from a Quality Computer by Oran Sands III.
Is IFF Really a Standard? by John Foust..
Amazing Stories and the Amiga™ by John Foust.
All about Printer Drivers by Richard Bielak
Intuition Gadgets by Harriet Maybeck Tolley.
Deluxe Video 1.2 by Bob Eller
Pro Video CG1 by Oran Sands III.
Digi-View 2.0 Digitizer/Software by Jennifer M. Janik
Prism HAM Editor from Impulse by Jennifer M. Janik
Easy! drawing tablet by John Foust...
CSA's Turbo-Amiga Tower by Alfred Aburto
68000 Assembly Language by Chris Martin...

Volume 2, Number 8

This month Amazing Computing focuses on entertainment packages for the Amiga. Amazing game reviews...

SDI, Earl Weaver Baseball, Portal!, The Surgeon,
Little Computer People, Sinbad, StarGlider,
King's Quest I,II and III, Feary Tale Adventure, Ultima III,
Facets of Adventure, Video Vegas and Bard's Tale.
Plus Amazing monthly columns...
Amiga Notes, Roomers, Modula-2, 68000 Assembly
Language and The Amicus Network.
Disk-2-Disk by Matthew Leeds
The ColorFonts Standard by John Foust
Skinny C Programs by Robert Riemersma, Jr.
Hidden Messages in Your Amiga™ by John Foust
The Consumer Electronics Show
and Comdex by J Foust

To Be Continued.....

•AC•

With a past like this....

The future is a tradition.

Since February 1986, **Amazing Computing**™ has been providing users with complete information for their Amigas. With the growth of the new Amiga 500 and Amiga 2000, the Amiga user needs more information than ever in selecting the right software and hardware for their needs.

Amazing Computing™ will continue to offer the Amiga user the best in technical knowledge and in unbiased reviews for the Commodore-Amiga™.

Amazing Computing™ will not rest on past achievements.

To Subscribe to Amazing Computing™ or to purchase Public Domain Software, please fill out the form below and send with a Check or Money Order to:

PiM Publications Inc.
P.O.Box 869
Fall River, MA 02722

Back issues are still available at $4.00 each (Foreign orders, please add $1.00 U.S. per issue for P. & H).
All payments must be made by check or money order in U.S. funds drawn on a U.S. Bank.

Amaze Me!

Please start my subscription to Amazing Computing™ with the next available issue. I have enclosed $24.00 for 12 issues in the U.S. ($30.00 Canada and Mexico, $35.00 overseas). All funds must be in U.S. Currency drawn on a U.S. Bank.

Please circle your selection:
Subscription PDS (as noted) Back Issues Sub Renewal

Name _____

Street _____

City _____ **St.** ____ **ZIP** _____

Amount enclosed _____

Mass. Residents, please add 5% sales tax on PDS orders

BACK ISSUES: $4.00 each (foreign orders add $1.00 each for Postage and Handling

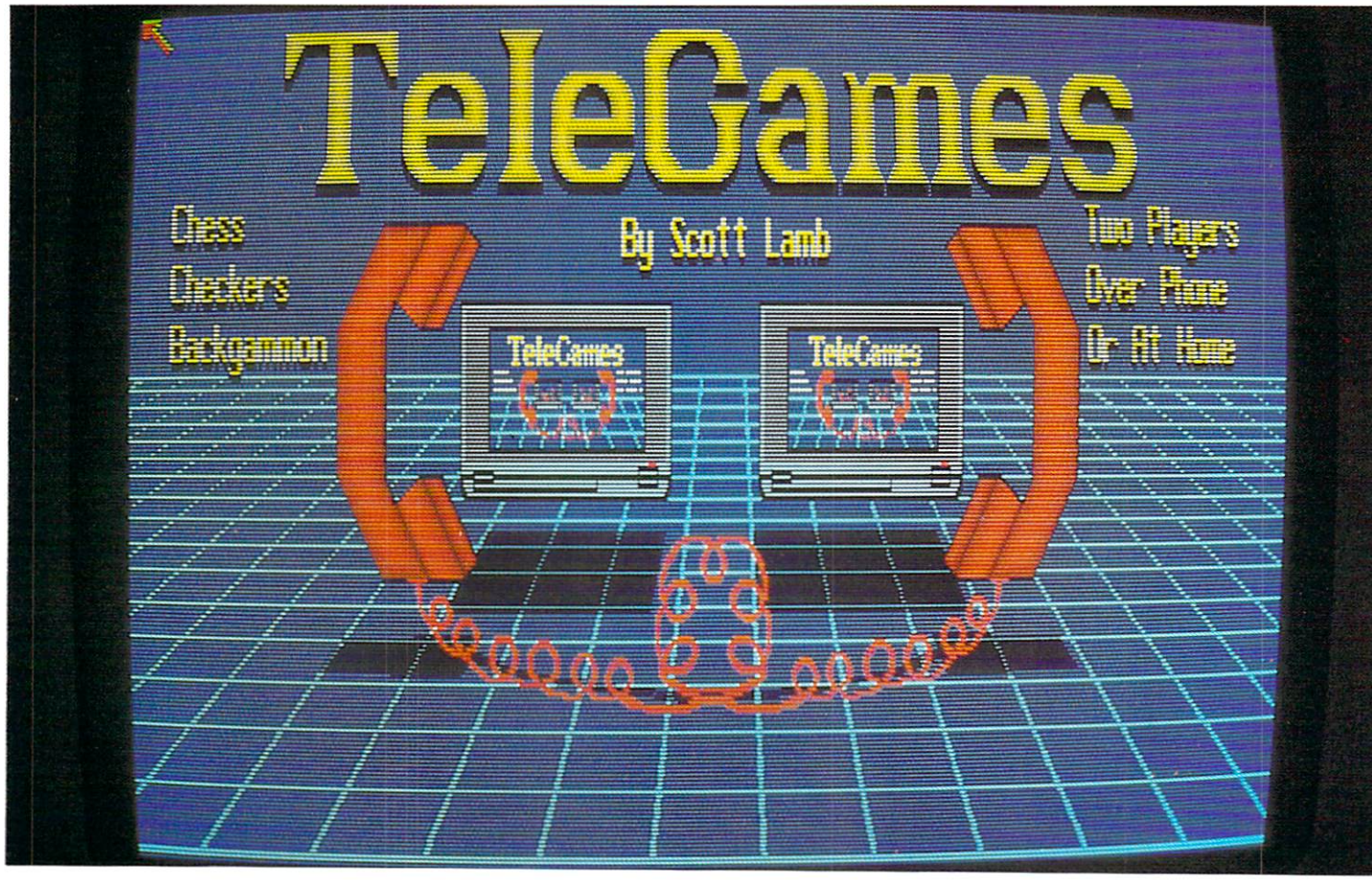
VOL.1#1 VOL.1#2 VOL.1#3 VOL.1#4 VOL.1#5 VOL.1#6
VOL.1#7 VOL.1#8 VOL.1#9 VOL.2#1 VOL.2#2 VOL.2#3
VOL.2#4 VOL.2#5 VOL.2#6 VOL.2#7 VOL.2#8

AC 9 87

Public Domain Software:
$6.00 each for subscribers (yes, even new ones!)
$7.00 each for non subscribers.

AMICUS: A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22

Fred Fish:
FF1 FF2 FF3 FF4 FF5 FF6 FF7 FF8 FF9 FF10
FF11 FF12 FF13 FF14 FF15 FF16 FF17 FF18 FF19 FF20
FF21 FF22 FF23 FF24 FF25 FF26 FF27 FF28 FF29 FF30
FF31 FF32 FF33 FF34 FF35 FF36 FF37 FF38 FF39 FF40
FF41 FF42 FF43 FF44 FF45 FF46 FF47 FF48 FF49 FF50
FF51 FF52 FF53 FF54 FF55 FF56 FF57 FF58 FF59 FF60
FF61 FF62 FF63 FF64 FF65 FF66 FF67 FF68 FF69 FF70
FF71 FF72 FF73 FF74 FF75 FF76 FF77 FF78 FF79 FFNA
FF81 FF82 FF83 FF84 FF85 FF86 FF87



TeleGames is what you've waited for. The Future is here.

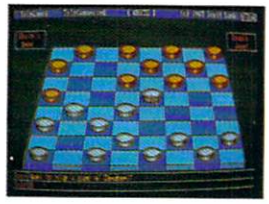
TeleGames allows you to use your computer and modem to play Chess, Checkers and Backgammon with a human opponent over the telephone. Only $34.95!

TeleGames Features

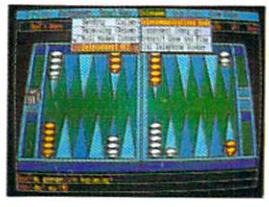
* Chess * Checkers * Backgammon
* Superb Graphic Game Simulations
* Smooth Depth Arranged Movement
* 4 angle 3D & 2D view perspectives
* Digitized Sound Effects
* Compatible with any modem
* 300, 1200, 2400, 9600 Baud
* Call originate or answer
* Null Modem Connect option
* Save Game & Transmit Game options
* Opponent File Directories
* Send and Receive Typed Messages
* Easy to Use Menus & Requesters
* All Official Game Rules Supported
* Play Over the Phone or at Home
* Legal Moves Graphically enacted on the TeleConnected computer
* Fully copyable to hard disks
* Upgrades available on our BBS

3D Chess

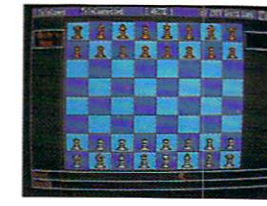
3D Checkers



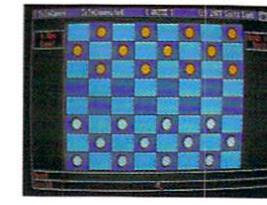
3D Backgammon



2D Chess



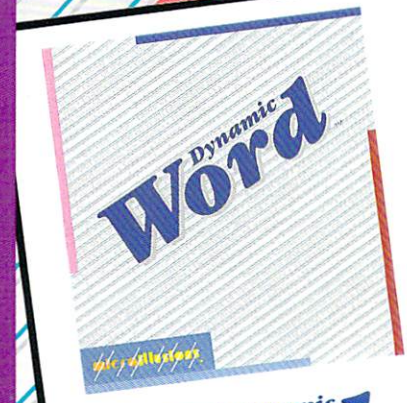
2D Checkers



2D Backgammon

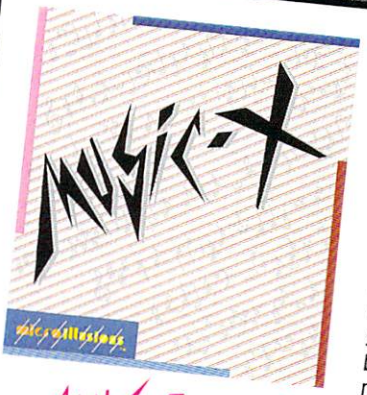
If you Enjoy Telecomputing, You'll Love TeleGames!

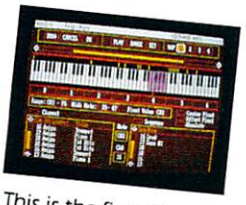
Published by Software Terminal
3014 Alta Mere, Fort Worth, TX 76116
817-244-4150 Modem: 817-244-4151
Dealer Inquiries Invited

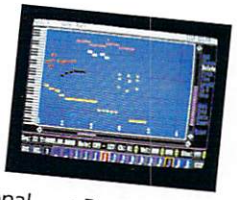


Dynamic Word™

This full featured word processor with speller and thesaurus is what Amiga users have all been waiting for. For those who have done serious comparisons, this is the software package for you! Just a few of the features Dynamic Word will provide you with are: Speller and Thesaurus • File compatibility with most systems • Multitasking and multiwindowing • Easy to use • Very fast operations • Supports multiple fonts and sizes • Desk top publishing type layout • Math calculations • Powerful screen editing mode for programmers • Macros • Full justification • On line help • Automatics • Table of contents and index generation. With all this and more, you will want to get your copy of Dynamic Word today!

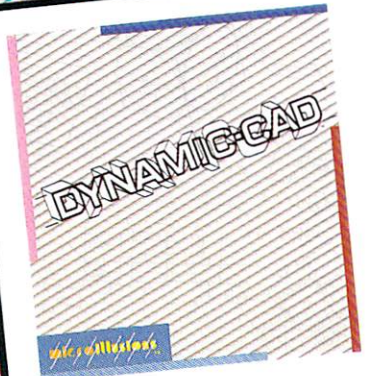






Music-X™

This is the first of many professional music products from Microillusions. Committed to pushing ahead the state of the art, Music X is a MIDI based music system with the power most professionals are used to on more expensive machines! Just a small sampling of the features you will find on Music X include: Librarian • Patch editors • Sequence editor • MIDI filtering • Up to 200 sequences • High resolution clock • Key mapping • Realtime/steptime, record and playback • Plus many more. Watch for Music X!



DYNAMIC-CAD™

The true premier cad product for the Amiga, Dynamic Cad 2.3 is dedicated to giving you the best with features like: Precision up to .0001" • Auto dimensioning • Schematic auto routing • Net listing • Parts list reporting • Gerber photoploter™ compatibility • Full plotter support: Calcomp,™ Houston Instruments,™ Roland, Hewlett Packard™ and many more, as well as dot matrix, ink jet and laser printers • Large symbol and pseudo libraries included • Full entry and edit functions • 8,192 levels to work on • Multiple resolution and color • Isometric presentation.
You can virtually do all types of cad applications like Architectural, Mechanical, PCB and Schematic Design with Dynamic Cad, pick up yours today!

OTHER PRODUCTS FROM MICROILLUSIONS

- **DYNAMIC PUBLISHER™**
  Watch for it in the Fall
- **MICRO MIDI™**
  Available soon
- **DYNAMIC-CAD™ ELECTRONIC DESIGNER**
  Watch for it in the Fall

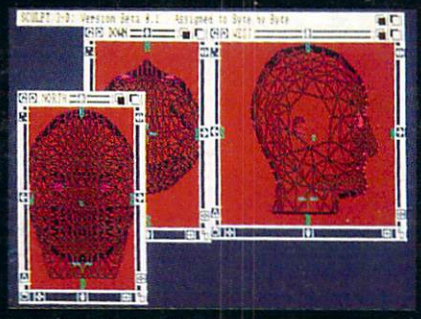
All of these products are now being developed for the Amiga, and will soon be available in other formats.

microillusions™

17408 Chatsworth St., Granada Hills, CA 91344, inside CA 818/360-3715 • outside CA 800/522-2041 • FAX 818/360-1464

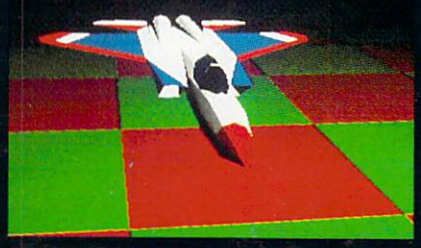
Create your own universe with

Sculpt-3D



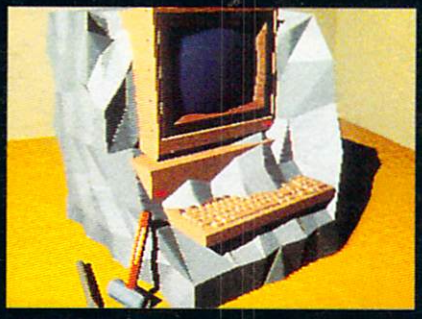
EASE OF USE

- Uses standard Amiga environment of mouse control, menus & windows
- Fully multi-tasking
- Wire frame edit mode
- Full library of graphics primitives
- Power tools like spin, grab, extrude, reflect and unslice



POWER

- Automatically handles shading, smoothing, anti-aliasing, shadows and texture
- Object files let you build your own libraries of primitives
- Overscan creates "true video" images
- Painting mode renders objects in seconds



CONTROL

- Complete control over the color, texture, smoothing and shading of every surface of every object
- Complete control of brightness, color and placement of multiple light sources
- Complete control over the camera position and angle of view

COMPATIBILITY

- All image files are IFF compatible
- Script file mode accepts ASCII input
- All resolution modes are supported
- Full support for video frame buffer boards and genlocks
- Not copy protected in any way shape or form

Look for ANIMATE 3-D coming in Summer '87

BYTE by BYTE™

Aboretum Plaza II 9442 Capital of Texas Highway North Suite 150 Austin, TX 78759 (512) 343-4357

Requirements: A1000, A500, or A2000 Amiga Computer with 512K RAM or more.